



Porting Manual : peak

특화 A507 하야코손

0. 목차

0. 목차

I. 개발 환경

1. 프로젝트 기술 스택
2. directory 구조
 - EC2 server
3. 설정 파일
 - Nginx, Jenkins, Redis : `docker-compose.yml`
 - Nginx : `nginx.conf`
 - Jenkins : `Jenkinsfile`
 - MongoDB : `docker-compose.yml`
 - Backend : `Dockerfile`
 - Frontend : `Dockerfile`

4. 환경 변수

- Backend : `application.yml`
- Frontend : `.env`

II. 데이터 처리

- twint 설치
- twint lib directory에 있는 `token.py` 파일을 아래 링크에 있는 대로 수정
- Selenium과 Chrome Browser 및 Driver를 이용한 crawling 환경 구현
- crawling library 설치

III. 빌드 및 배포

1. docker 설치
2. Docker Compose 설치
3. SSL 인증
 - certbot docker container
4. DB 및 Infra 배포
 - Nginx, Jenkins, Redis
 - MongoDB
 - Dockerfile 저장
 - 방법 1) 수동 배포
 - 방법 2) Jenkins 사용

IV. 외부 서비스

1. Kakao OAuth2.0
2. YouTube Data API
3. Google Elastic Search

I. 개발 환경

1. 프로젝트 기술 스택

- CI/CD
 - AWS EC2
 - Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-1018-aws x86_64)
 - Docker 20.10.23
 - Jenkins 2.375.2
 - Nginx 1.23.3
- Backend
 - IntelliJ IDEA 2022.3.1
 - JVM OpenJDK 11
 - SpringBoot Gradle 2.7.10
 - Spring Security
 - Spring Data MongoDB
 - JWT 0.11.5
 - Swagger
- Frontend
 - Visual Studio Code 1.77.1
 - Node.js 16.16.0
 - TypeScript 4.9.5
 - React 18.2.0
 - Redux
 - Apache ECharts 5.4.2
- Database
 - MongoDB 6.0.5
 - Redis 7.0.10

2. directory 구조

▼ EC2 server

```
/home/ubuntu
├── crawler
│   ├── __pycache__
│   ├── chromedriver
│   ├── last_crawled_url_list.txt
│   ├── main.py
│   ├── news_crawler.py
│   ├── news_index.txt
│   ├── stderr.txt
│   ├── stdin.txt
│   ├── stdout.txt
│   ├── test.py
│   ├── twitter_crawler.py
│   └── twitter_crawling.json
├── idol_images
│   ├── BTS.webp
│   ├── 세븐틴.webp
│   ├── ...
│   ├── ...
│   └── NewJeans.webp
```

```

├─ jenkins
│  └─ logs
│     └─ workspace
│        └─ peak
├─ miniconda3
├─ mongo
│  └─ docker-compose.yml
├─ nginx
│  └─ conf.d
│     └─ nginx.conf
├─ redis
│  └─ conf
│     └─ redis.conf
│  └─ data
├─ watcher
│  └─ analyzed
│     └─ news
│        └─ IN
│           └─ NK
│              └─ TK
│                 └─ rank
│                    └─ RD
│                       └─ RH
│                          └─ twitter
│                             └─ PD
├─ crawled
│  └─ news
│     └─ twitter
├─ pem_puttygen
├─ stderr.txt
├─ stdin.txt
├─ stdout.txt
├─ watch_sftp.py
└─ docker-compose.yml

```

3. 설정 파일

Nginx, Jenkins, Redis : docker-compose.yml

~/docker-compose.yml

```

version: '3.8'

services:
  nginx:
    image: nginx
    container_name: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - ~/nginx/conf.d/nginx.conf:/etc/nginx/conf.d/default.conf
      - /etc/letsencrypt:/etc/letsencrypt
      - ~/idol_images:/var/www/images/
    command:
      ['nginx', '-g', 'daemon off;']

  jenkins:
    image: jenkins/jenkins
    container_name: jenkins
    volumes:
      - /usr/bin/docker:/usr/bin/docker
      - /var/run/docker.sock:/var/run/docker.sock
      - ~/jenkins:/var/jenkins_home
    ports:
      - 8095:8080
    privileged: true
    user: [계정명]
    restart: unless-stopped

  redis:
    image: redis
    container_name: redis
    ports:
      - 6379:6379
    command: /bin/sh -c "redis-server --requirepass [비밀번호]"
    volumes:
      - ~/redis/data:/data
      - ~/redis/conf/redis.conf:/usr/local/etc/redis/redis.conf
    environment:

```

```
- TZ=Asia/Seoul
restart: always
network_mode: host
```

Nginx : nginx.conf

~/nginx/conf.d/ `nginx.conf`

```
server {
    listen 80;
    listen [::]:80;
    server_name [서비스 도메인];

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

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name [서비스 도메인];

    ssl_certificate /etc/letsencrypt/live/[서비스 도메인]/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/[서비스 도메인]/privkey.pem;
    #ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
    ssl_prefer_server_ciphers on;
    #include /etc/letsencrypt/options-ssl-nginx.conf;

    proxy_set_header Host $host;
    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 X-Forwarded-Proto https;
    #proxy_set_header Upgrade $http_upgrade;
    #proxy_set_header Connection "Upgrade";
    proxy_headers_hash_bucket_size 512;
    proxy_redirect off;

    location / {
        proxy_pass http://[서비스 도메인]:3126/;
    }
    location /api/ {
        # add_header 'Access-Control-Allow-Origin' '*';
        proxy_pass http://[서비스 도메인]:8093/;
    }
    location /img/ {
        alias /var/www/images/;
        autoindex on;
    }
}
```

Jenkins : Jenkinsfile

~/jenkins/workspace/peak/ `Jenkinsfile`

```
pipeline {
    agent any

    tools {
        nodejs 'nodejs-16.16.0'
    }

    stages {
        stage('Gradle Build') {
            steps {
                dir("back") {
                    sh 'chmod +x gradlew'
                    sh './gradlew clean build -x test'
                }
            }
        }
        stage('React Build') {
            steps {
                dir("front") {
```

```

        sh 'npm install --force'
        sh 'npm run build'
    }
}

stage('Backend Docker Build') {
    steps {
        dir("back") {
            sh 'docker build -t peak-backend:latest .'
        }
    }
}

stage('Frontend Docker Build') {
    steps {
        dir("front") {
            sh 'docker build -t peak-frontend:latest .'
        }
    }
}

stage('Backend Deploy') {
    steps {
        sh 'docker rm -f backend'
        sh 'docker run -d --name backend -p 8093:8080 -u root peak-backend:latest'
    }
}

stage('Frontend Deploy') {
    steps {
        sh 'docker rm -f frontend'
        sh 'docker run -d --name frontend -p 3126:3000 -u root peak-frontend:latest'
    }
}

stage('Finish') {
    steps {
        sh 'docker images -qf dangling=true | xargs -I{} docker rmi {}'
    }
}
}
}

```

MongoDB : docker-compose.yml

~/mongo/ `docker-compose.yml`

```

version: "3"

services:
  mongodb:
    image: mongo
    restart: always
    container_name: mongo
    ports:
      - "27017:27017"
    environment:
      MONGO_INITDB_ROOT_USERNAME: [계정명]
      MONGO_INITDB_ROOT_PASSWORD: [비밀번호]
    volumes:
      - /data/mongodb/data/db:/data/db

```

Backend : Dockerfile

~/jenkins/workspace/peak/back/ `Dockerfile`

```

FROM openjdk:11-jre-slim

ARG JAR_FILE=build/libs/peak-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar

EXPOSE 8093
ENTRYPOINT ["java", "-jar", "app.jar"]

```

Frontend : Dockerfile

~/jenkins/workspace/peak/front/ `Dockerfile`

```
FROM node:16
WORKDIR /app
RUN npm install -g serve

RUN mkdir ./build
COPY ./build ./build

EXPOSE 3126
ENTRYPOINT ["serve", "-s", "build"]
```

4. 환경 변수

Backend : `application.yml`

~/jenkins/workspace/peak/back/src/main/resources/ `application.yml`

```
serverHost: [서비스 도메인]
baseUrl: [api 요청 경로 (서버)]
redirectUrl: [redirect url (클라이언트)]

server:
  port: 8080
  servlet:
    encoding:
      charset: UTF-8
      enabled: true
      force: true
spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: [카카오 REST API KEY]
            client-secret: [카카오 SECRET KEY]
            redirect-uri: '${baseUrl}/login/oauth2/code/kakao'
            scope:
              - account_email
            authorization-grant-type: authorization_code
            client-authentication-method: POST
            client-name: Kakao
        provider:
          kakao:
            authorization-uri: https://kauth.kakao.com/oauth/authorize
            token-uri: https://kauth.kakao.com/oauth/token
            user-info-uri: https://kapi.kakao.com/v2/user/me
            user-name-attribute: id
      data:
        mongodb:
          host: ${serverHost}
          port: 27017
          authentication-database: admin
          database: [Database 명]
          username: [계정 명]
          password: [비밀번호]
          field-naming-strategy: org.springframework.data.mapping.model.SnakeCaseFieldNamingStrategy
        redis:
          host: ${serverHost}
          port: 6379
          password: [비밀번호]
          lettuce:
            pool:
              max-active: 10
              max-idle: 10
              min-idle: 2
      jwt:
        header: Authorization
        secret: [비밀 키]
        access-token-valid-time: 1800
        refresh-token-valid-time: 604800
      logging:
        level:
          org.springframework.data.mongodb.core.MongoTemplate: DEBUG
      springdoc:
```

```

swagger-ui:
  path: /swagger-ui.html
  display-request-duration: true
  groups-order: DESC
  operationsSorter: method
  disable-swagger-default-url: true
youtube:
  api:
    key: [YouTube Data API 키]
    application: google-youtube-api-search
  search:
    type: video
    fields: items(id/videoId,snippet/title,snippet/thumbnails/high/url)

```

Frontend : .env

~/jenkins/workspace/peak/front/ .env

```

REACT_APP_GOOGLE_ANALYTICS_TRACKING_ID=[Google Analytics 추적 Id]
REACT_APP_YOUTUBE_KEY=[YouTube Data API 키]
REACT_APP_BASE_URL=[서비스 도메인]

```

II. 데이터 처리


twint 설치

```
$ pip3 install --user --upgrade git+https://github.com/twintproject/twint.git@origin/master#egg=twint
```

twint lib directory에 있는 token.py 파일을 아래 링크에 있는 대로 수정

This is one to modify twint which is python module to scrape twitter without API token. In order to use, you must put this script on `twint/` and replace

This is one to modify twint which is python module to scrape twitter without API token. In order to use, you must put this script on `twint/` and replace default one. - GitHub

 <https://gist.github.com/moxak/ed83dd4169112a0b1669500fe855101a>

Selenium과 Chrome Browser 및 Driver를 이용한 crawling 환경 구현

```

$ wget https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
$ sudo apt install ./google-chrome-stable_current_amd64.deb
$ google-chrome --version
$ wget https://chromedriver.storage.googleapis.com/111.0.5563.64/chromedriver_linux64.zip
$ sudo apt-get install unzip
$ unzip chromedriver_linux64.zip

```

crawling library 설치

```

$ pip install shutil
$ pip install selenium
$ pip install asyncio
$ pip install nest-asyncio

```

III. 빌드 및 배포

▼ 1. docker 설치

apt-get update, 관련 package 설치

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

Docker 공식 GPG-Key 추가

```
sudo mkdir -m 0755 -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

Repository 설정

```
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
```

2. Docker Compose 설치

Install the Compose standalone

How to install Docker Compose - Other Scenarios

 <https://docs.docker.com/compose/install/other/>



1. To download and install Compose standalone, run :

```
sudo curl -SL \
https://github.com/docker/compose/releases/download/v2.16.0/docker-compose-linux-x86_64 \
-o /usr/local/bin/docker-compose
```

2. Apply executable permissions to the standalone binary in the target path for the installation.

실행 권한 설정

```
sudo chmod +x /usr/local/bin/docker-compose
```

- If the command `docker-compose` fails after installation, check your path. You can also create a symbolic link to `/usr/bin` or any other directory in your path. For example:

심볼릭 링크 설정

```
sudo ln -s /usr/local/bin/docker-compose /usr/bin/docker-compose
```

- 정상 설치 확인


```
docker-compose -v
```

3. SSL 인증

certbot docker container

```
docker run -it --rm --name certbot \
-p 80:80 \
-v '/etc/letsencrypt:/etc/letsencrypt' \
-v '/var/lib/letsencrypt:/var/lib/letsencrypt' \
certbot/certbot certonly -d 'j8a507.p.ssafy.io' --standalone \
--server https://acme-v02.api.letsencrypt.org/directory
```

4. DB 및 Infra 배포

Nginx, Jenkins, Redis

```
# docker-compose.yml 파일의 위치에서 실행 (현재 home directory)
cd /home/ubuntu 또는 cd ~
docker compose up -d
```

MongoDB

```
# docker-compose.yml 파일의 위치에서 실행 (현재 mongo directory)
cd /home/ubuntu/mongo 또는 ~/mongo
docker compose up -d
```

Dockerfile 저장

- Backend
 - 프로젝트 back 폴더 최상위에 Backend Dockerfile을 위치시키고 Gitlab Repository에 push
- Frontend
 - 프로젝트 front 폴더 최상위에 Frontend Dockerfile을 위치시키고 Gitlab Repository에 push

방법 1) 수동 배포

ec2 서버에서 git pull 받은 후, 각각 directory에 진입해서 다음 명령어 실행 (각 Dockerfile의 위치)

- back

```
# ~/[GitLab Repository 명]/back/

chmod +x gradlew
./gradlew clean build -x test
docker build -t peak-backend:latest .
docker rm -f backend
docker run -d --name backend -p 8093:8080 -u root peak-backend:latest
docker images -qf dangling=true | xargs -I{} docker rmi {}
```

- front

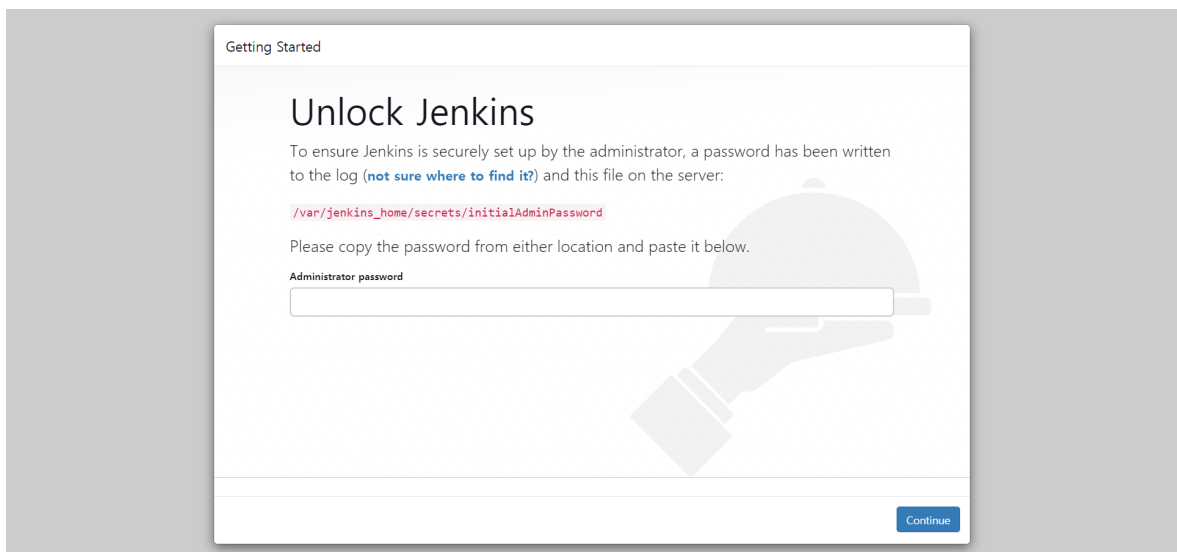
```
# ~/[GitLab Repository 명]/front/

npm install --force
npm run build
docker build -t peak-frontend:latest .
docker rm -f frontend
docker run -d --name frontend -p 3126:3000 -u root peak-frontend:latest
docker images -qf dangling=true | xargs -I{} docker rmi {}
```

▼ 방법 2) Jenkins 사용

Jenkins (1)

1. http://[domain]:[jenkins_port] 접속



- **Administrator password** : ec2서버에서 **docker logs [jenkins docker container name]** 입력 후 나오는 password를 입력

```

*****
*****
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

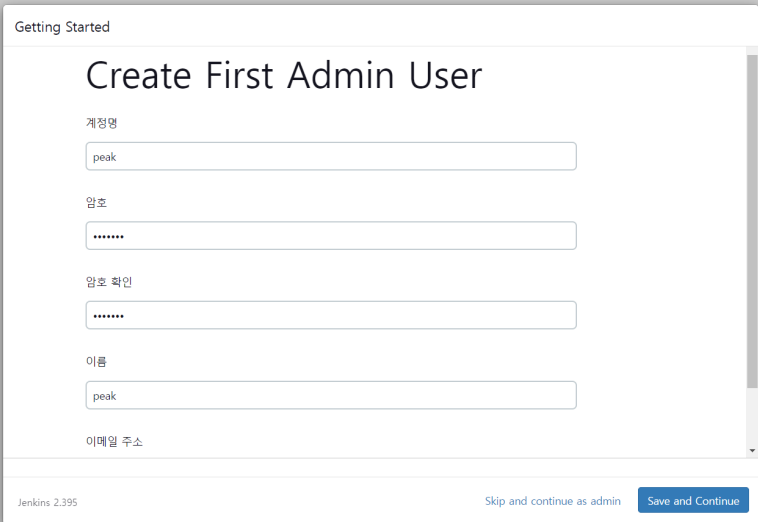
#####Administrator password#####

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*****
*****
*****

```

2. Create First Admin User 계정 등록



The screenshot shows the 'Getting Started' page for Jenkins 2.395, specifically the 'Create First Admin User' form. The form contains the following fields:

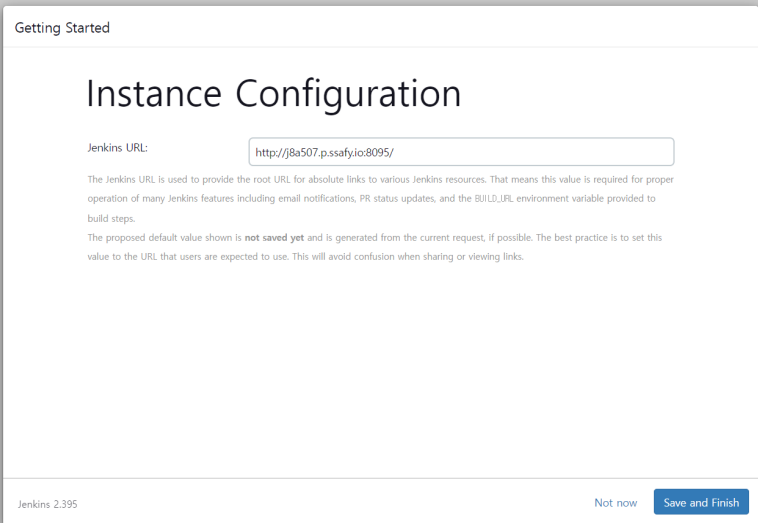
- 계정명 (Username):** peak
- 암호 (Password):** [masked with dots]
- 암호 확인 (Confirm Password):** [masked with dots]
- 이름 (Name):** peak
- 이메일 주소 (Email Address):** [empty]

At the bottom right, there are two buttons: 'Skip and continue as admin' and 'Save and Continue'.

- 계정명 : 임의 작성
- 암호 : 임의 작성
- 암호 확인 : 임의 작성
- 이름 : 임의 작성
- 이메일 주소 : 임의 작성

⇒ Save and Continue

3. Instance Configuration



The screenshot shows the 'Getting Started' page for Jenkins 2.395, specifically the 'Instance Configuration' form. The form contains the following field:

- Jenkins URL:** http://8a507.p.ssafy.io:8095/

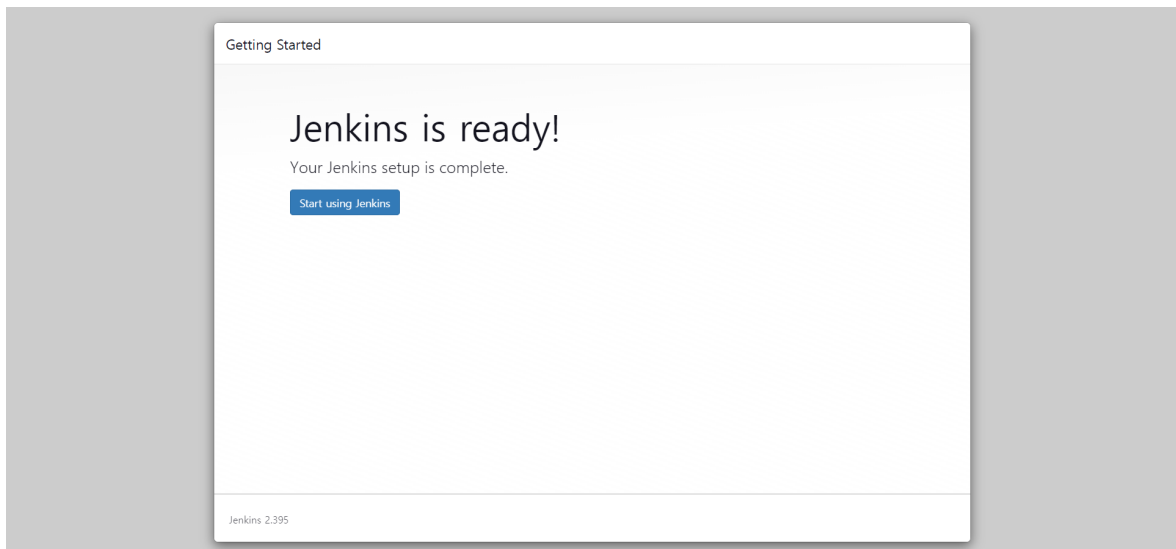
Below the field, there is explanatory text: 'The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps. The proposed default value shown is not saved yet and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.'

At the bottom right, there are two buttons: 'Not now' and 'Save and Finish'.

- Jenkins URL : 현재 접속한 URL (http://[domain]:[jenkins_port]) 입력

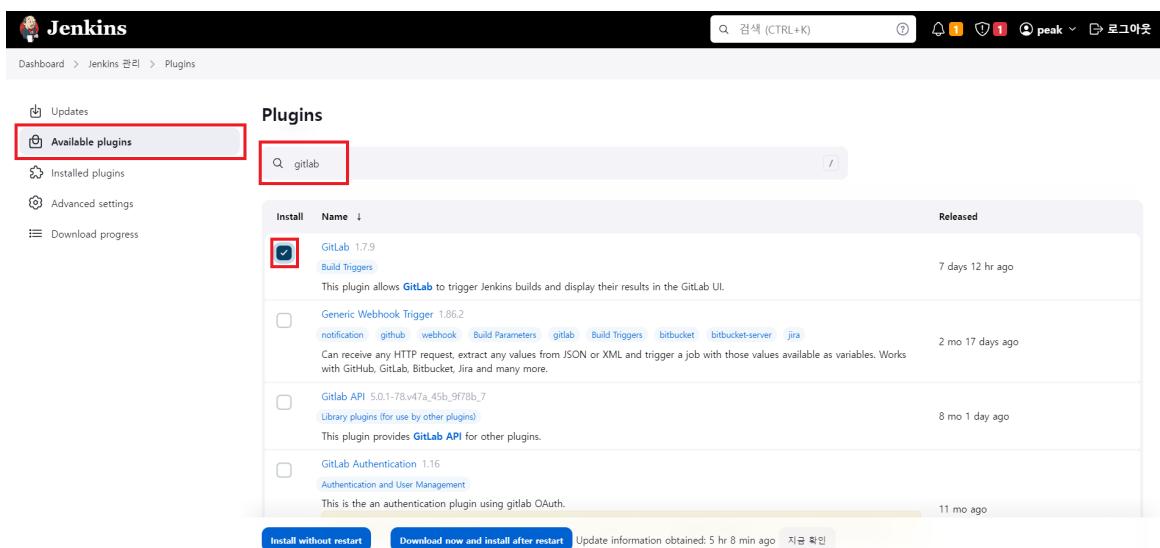
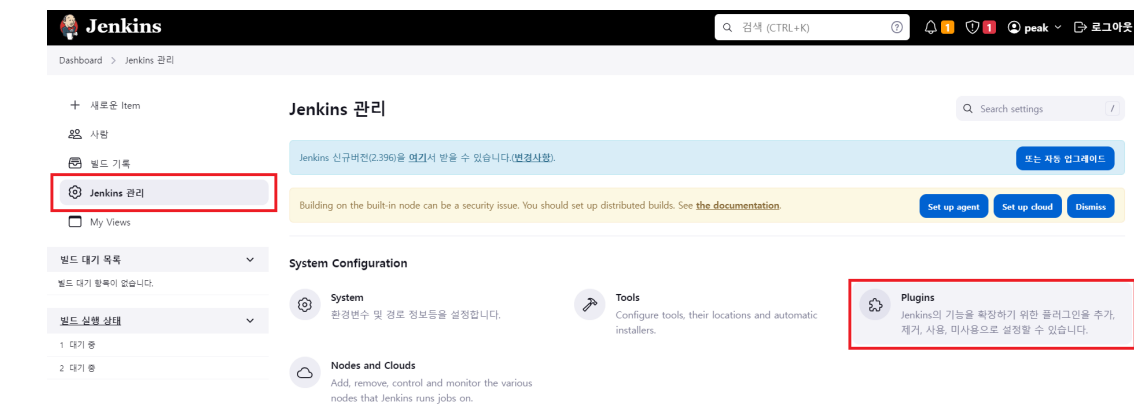
⇒ Save and Finish

4. Jenkins is ready!



⇒ Start using jenkins

5. Dashboard > Jenkins 관리 > Plugins > Available plugins



Jenkins

Dashboard > Jenkins 관리 > Plugins

Updates

Available plugins

Installed plugins

Advanced settings

Download progress

Plugins

nodejs

Install	Name	Released
<input checked="" type="checkbox"/>	GitLab 1.7.9 Build Triggers This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.	7 days 12 hr ago
<input checked="" type="checkbox"/>	NodeJS 1.6.0 npm NodeJS Plugin executes NodeJS script as a build step.	1 mo 18 days ago

[Install without restart](#)
[Download now and install after restart](#)
 Update information obtained: 5 hr 8 min ago
 [지금 확인](#)

→ [메인 페이지로 돌아가기](#)
(설치된 플러그인을 바로 사용할 수 있습니다.)

→ ☒ 설치가 끝나고 실행중인 작업이 없으면 Jenkins 재시작.

⇒ [Gibit lab](#), [NodeJS](#) 검색하여 추가

6. Dashboard > Jenkins 관리 > Tools

Dashboard > Jenkins 관리

+ 새로운 Item

사람

빌드 기록

프로젝트 연관 관계

파일 핑거프린트 확인

Jenkins 관리

My Views

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

Jenkins 관리

Jenkins 신규버전(2.396)을 [여기서](#) 받을 수 있습니다.([변경사항](#)).

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

System Configuration

System
환경변수 및 경로 정보등을 설정합니다.

Nodes and Clouds
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Tools
Configure tools, their locations and automatic installers.

Managed files
e.g. settings.xml for maven, central managed scripts, custom files, ...

NodeJS

Dashboard > Jenkins 관리 > Tools

NodeJS

NodeJS installations
List of NodeJS installations on this system

[Add NodeJS](#)

NodeJS

Name
nodejs-16.16.0

☒ Install automatically ?

Install from nodejs.org

Version
NodeJS 16.16.0

For the underlying architecture, if available, force the installation

☐ Force 32bit architecture

Global npm packages to install
Specify list of packages to install globally -- see npm install -g.

[Save](#) [Apply](#)

- Name : 임의 설정
- Version : **NodeJS 16.16.0** (사용할 버전 임의 선택)

⇒ Save

7. Dashboard > Jenkins 관리 > Manage Credentials

Jenkins

Dashboard >

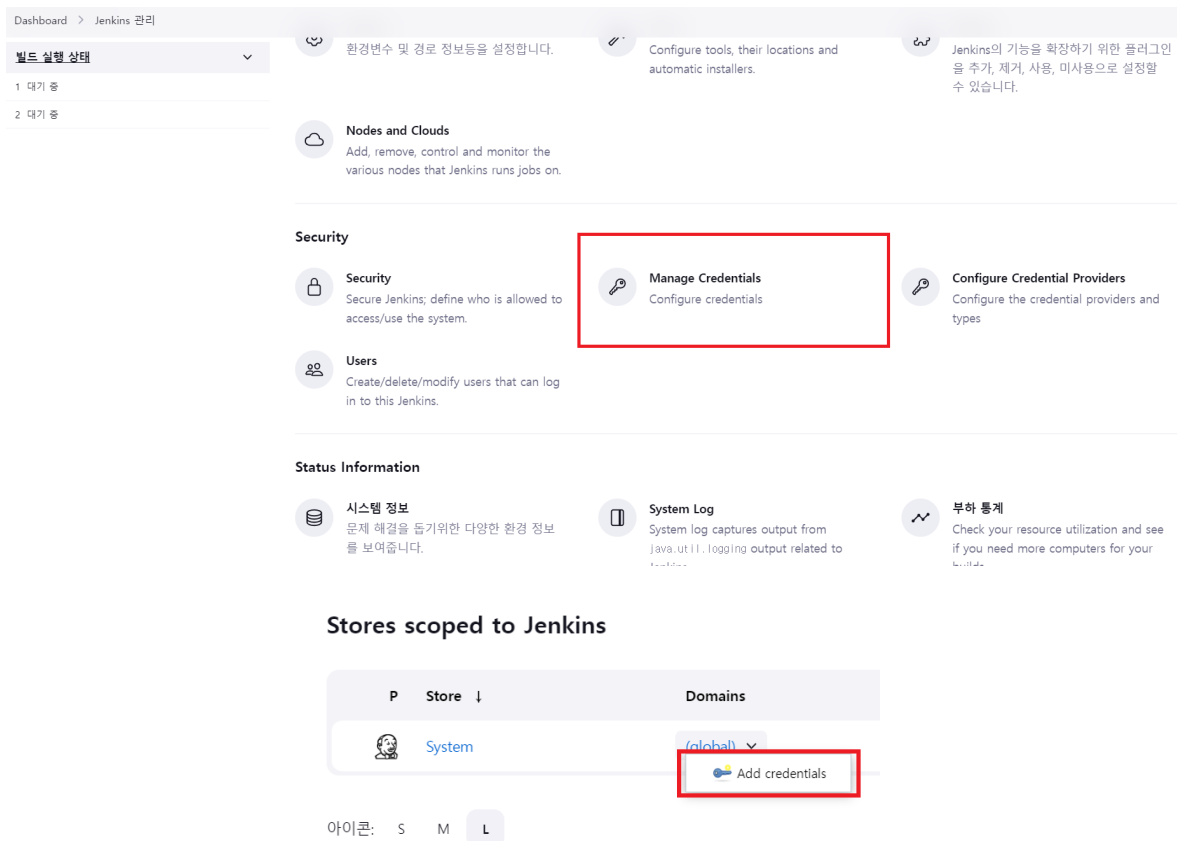
- + 새로운 Item
- 사람
- 빌드 기록
- Jenkins 관리**
- My Views

빌드 대기 목록 ▼

빌드 대기 항목이 없습니다.

빌드 실행 상태 ▼

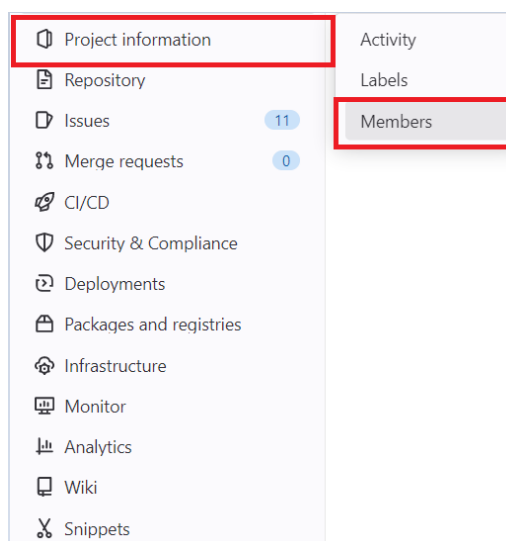
- 1 대기 중
- 2 대기 중



- (global)에 커서를 대면 나오는 우측 화살표 버튼을 눌러 **Add Credentials** 클릭

Gitlab (1)

8. Project infomagion > Members



Max role

Developer ▾

Maintainer ▾

Developer ▾

Maintainer ▾

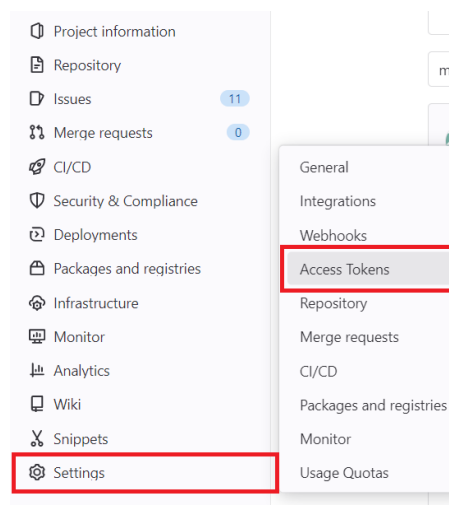
Developer ▾

Developer ▾

- Maintainer 권한 확인

⇒ 권한이 Maintainer가 아닐 경우 팀장(Maintainer권한자)에게 권한 변경 요청
또는 Maintainer가 직접 진행

9. Settings > Access Tokens



Q Search page

Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API.
You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

Add a project access token

Enter the name of your application, and we'll return a unique project access token.

Token name
peak-access-token

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

Expiration date
2023-04-30

Select a role
Maintainer

Select scopes
Scopes set the permission levels granted to the token. [Learn more.](#)

☐ api
Grants complete read and write access to the scoped project API, including the Package Registry.

☒ read_api
Grants read access to the scoped project API, including the Package Registry.

☒ read_repository
Grants read access (pull) to the repository.

☐ write_repository
Grants read and write access (pull and push) to the repository.

Create project access token

- Token name : 임의 설정
- Expiration date : 임의 설정 (프로젝트 기간보다 조금 여유를 두고 설정하였음)
- Select a role : Maintainer
- Select scopes : 임의 설정 (✓ read_api, ✓ read_repository 체크)

⇒ Create project access token

📘 Your new project access token has been created. X

Q Search page

Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API.
You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

Your new project access token

Make sure you save it - you won't be able to access it again.

Add a project access token
Enter the name of your application, and we'll return a unique project access token.

Copy project access token

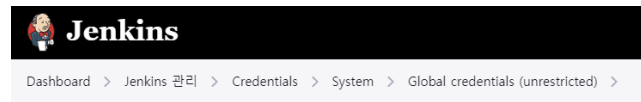
- access token을 발급받은 후, 최상단에서 Copy project access token

Jenkins (2)

10. 7 에서 Add Credentials 클릭 후 화면

Kind

- Username with password
- Username with password
- GitHub App
- GitLab API token
- SSH Username with private key
- Secret file
- Secret text
- Certificate



New credentials

Kind

GitLab API token

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

API token

.....

ID ?

peak

Description ?

Create

- Kind : GitLab API token 선택
- Scope : Global (Jenkins, nodes, items, all child items, etc) 선택
- API token : 9 에서 발급받아 Copy했던 Gitlab의 project access token 입력

11. Dashboard > Jenkins 관리 > System > Gitlab

Dashboard > Jenkins 관리

새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

Jenkins 관리

Jenkins 신규버전(2.396)을 [여기서](#) 받을 수 있습니다.(변경사항).

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

System Configuration

System

환경변수 및 경로 정보등을 설정합니다.

Tools

Configure tools, their locations and automatic installers.

Nodes and Clouds

Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Managed files

e.g. settings.xml for maven, central managed scripts, custom files, ...

Dashboard > Jenkins 관리 > System >

Gitlab

☒ Enable authentication for '/project' end-point

GitLab connections

Connection name

A name for the connection

peak-repository

Gitlab host URL

The complete URL to the Gitlab server (e.g. http://gitlab.mydomain.com)

https://lab.ssafy.com/

Credentials

API Token for accessing Gitlab

GitLab API token

Add +

고급 ▾

저장

Apply

- Connection name : 임의 설정
- Gitlab host URL : 깃랩 호스트 입력
- Credentials : GitLab API token 선택 (10 에서 생성한 Credentials)

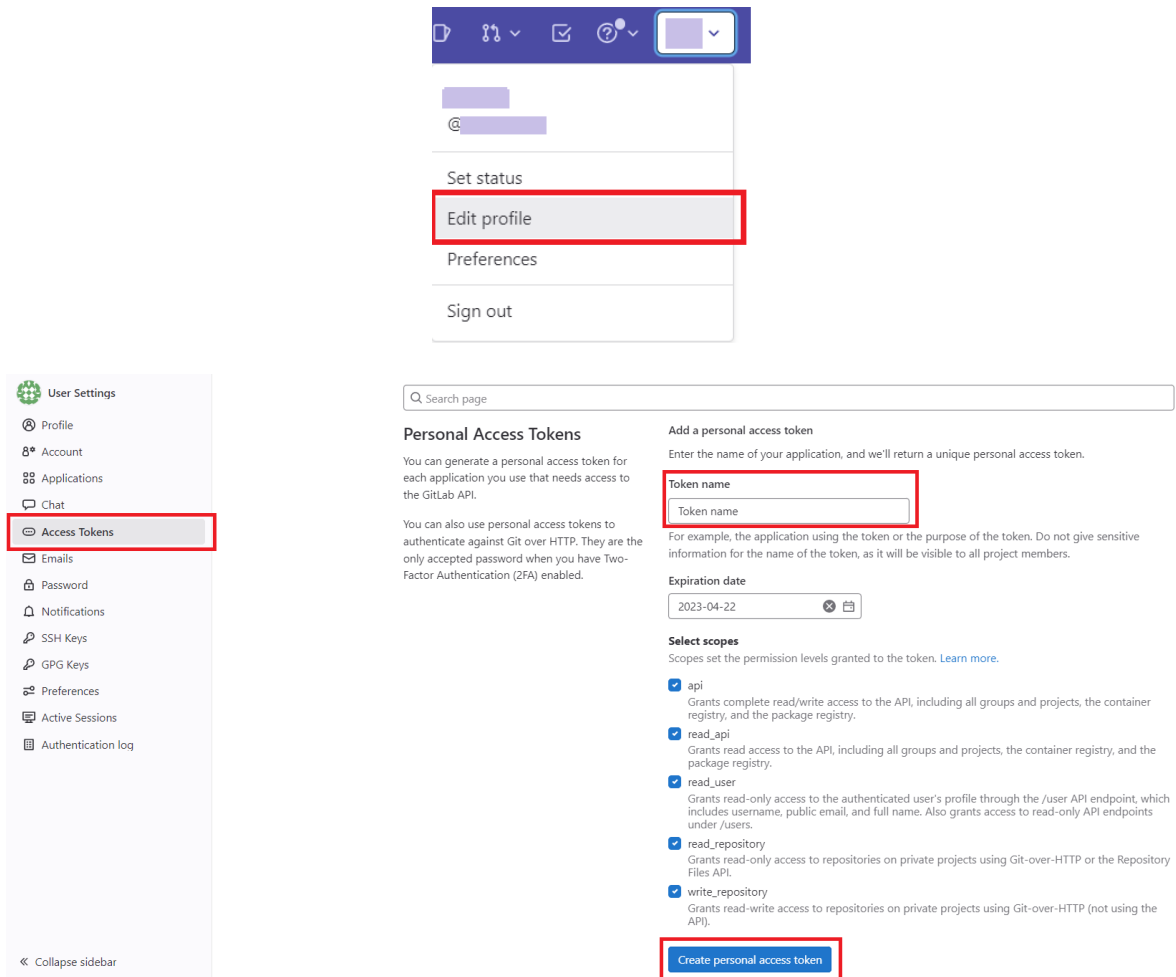
⇒ 저장

Gitlab (2)

12. 우측 상단 Profile > Edit profile > Access Tokens

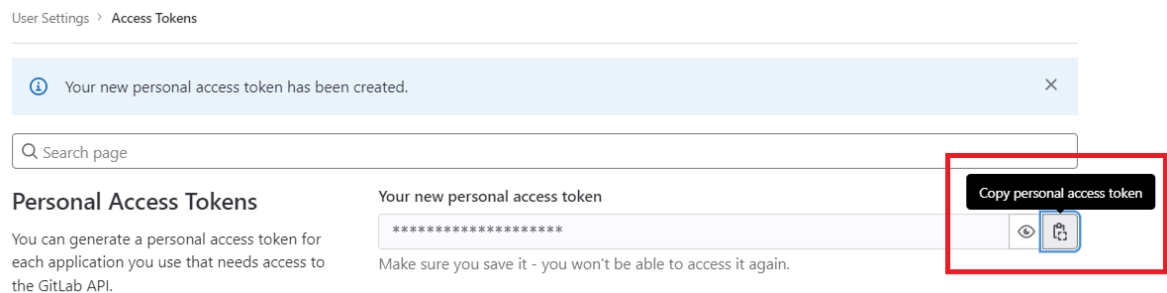
Porting Manual : peak

19



- Token name : 임의 설정
- Select scopes : 임의 설정 (✓ all 체크)

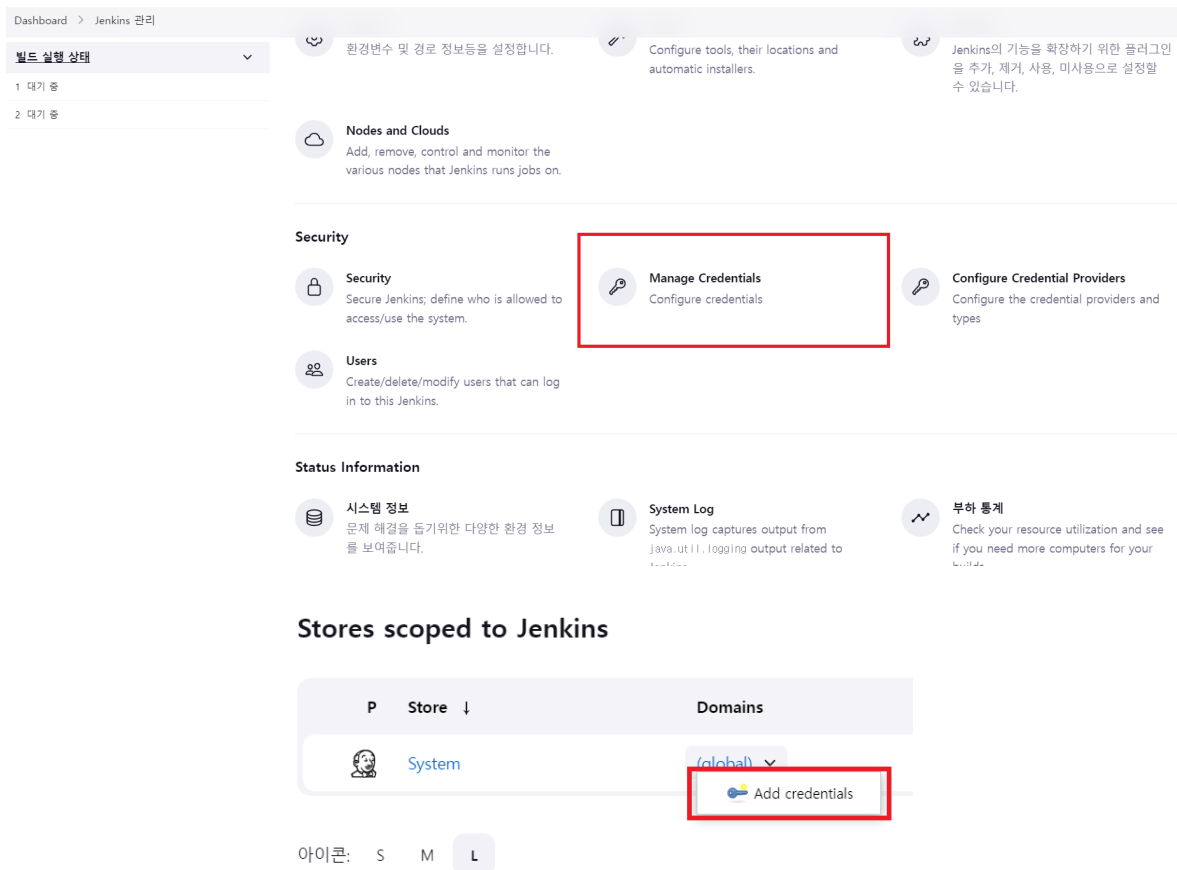
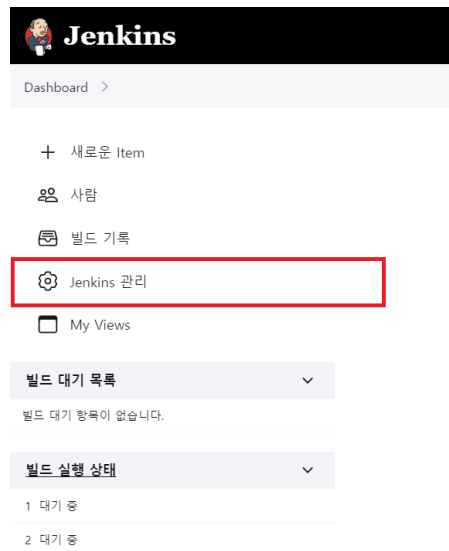
⇒ Create personal access token



- access token을 발급받은 후, 최상단에서 Copy personal access token

Jenkins (3)

13. Dashboard > Jenkins 관리 > Manage Credentials



- (global)에 커서를 대면 나오는 우측 화살표 버튼을 눌러 **Add Credentials** 클릭

New credentials

Kind

Username with password

Scope ?

Global (Jenkins, nodes, items, all child items, etc)

Username ?

Treat username as secret ?

Password ?

ID ?

Description ?

Create

- Kind : Username with password 선택
- Scope : Global (Jenkins, nodes, items, all child items, etc) 선택
- Username : Gitlab ID (personal token을 발급받은 사용자)
- Password : Gitlab Password (12 에서 발급받아 Copy했던 Gitlab의 personal access token 입력)
- ID : 임의 설정

14. Dashboard > 새로운 Item

Jenkins

Dashboard >

+ 새로운 Item

사람

빌드 기록

Jenkins 관리

My Views

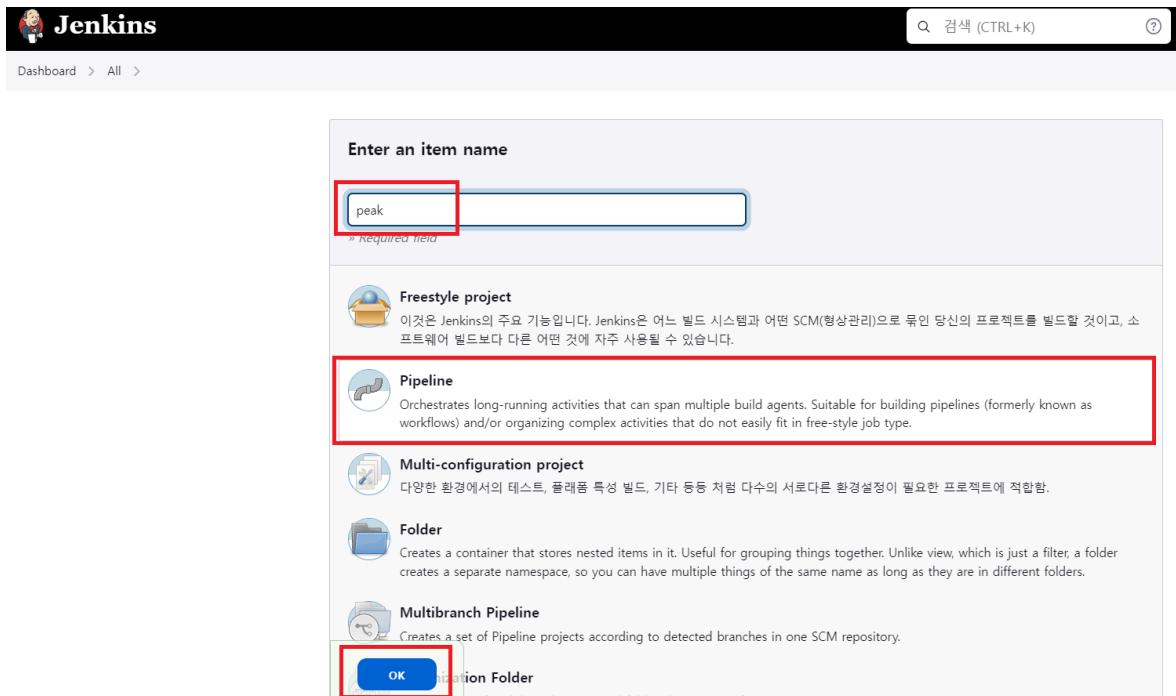
빌드 대기 목록

빌드 대기 항목이 없습니다.

빌드 실행 상태

1 대기 중

2 대기 중

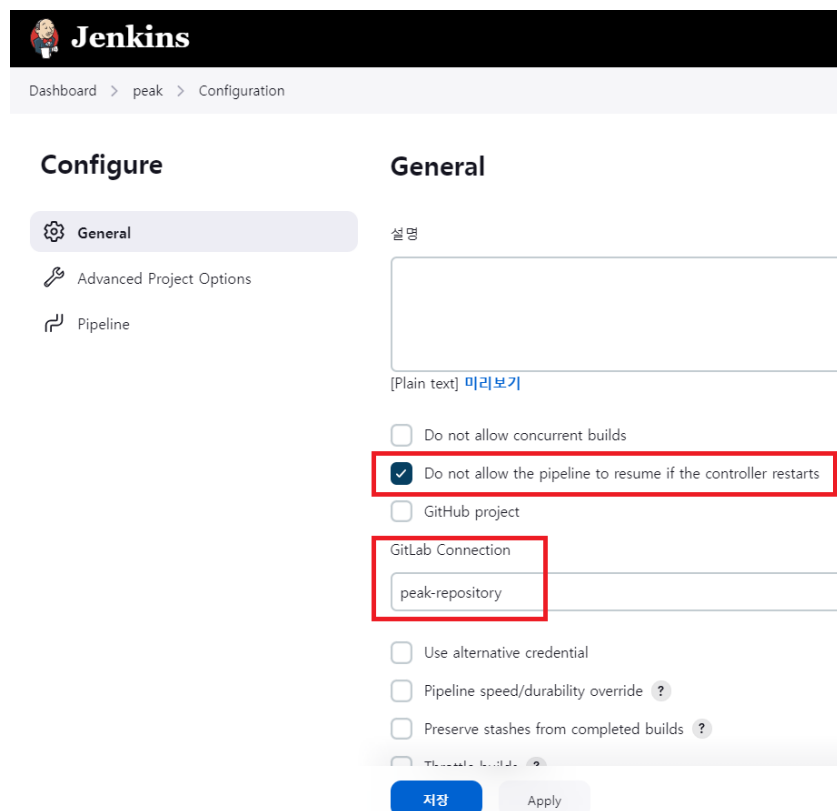


The image shows the Jenkins 'Enter an item name' screen. At the top, there's a search bar with '검색 (CTRL+K)' and a help icon. Below the header, the breadcrumb is 'Dashboard > All >'. The main form has a text input field containing 'peak', which is highlighted with a red box. Below the input field, there's a list of project types: 'Freestyle project', 'Pipeline' (highlighted with a red box), 'Multi-configuration project', 'Folder', and 'Multibranch Pipeline'. At the bottom, there's a blue 'OK' button, also highlighted with a red box.

- Enter an item name : 임의 설정 (프로젝트 명 입력)
- Pipeline 선택

⇒ OK

15. OK 클릭 시 나오는 화면 (Dashboard > peak > Configuration) > General



The image shows the Jenkins 'Configuration' screen for the 'peak' project, specifically the 'General' tab. The breadcrumb is 'Dashboard > peak > Configuration'. On the left, there's a 'Configure' section with tabs for 'General' (selected), 'Advanced Project Options', and 'Pipeline'. The main area is titled 'General' and contains a description field, a checkbox for 'Do not allow concurrent builds', a checked checkbox for 'Do not allow the pipeline to resume if the controller restarts' (highlighted with a red box), a checkbox for 'GitHub project', and a 'GitLab Connection' section with a text input field containing 'peak-repository' (highlighted with a red box). At the bottom, there are buttons for '저장' (Save) and 'Apply'.

- ☒ Do not allow the pipeline to resume if the controller restarts 체크
- GitLab Connection : 11 에서 설정했던 Gitlab Connection name 선택

Dashboard > peak > Configuration

Configure

- General**
- Advanced Project Options
- Pipeline

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: ?

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

- ☒ Approved Merge Requests (EE-only)
- ☒ Comments

Build Triggers

- ☒ Build when a change is pushed to GitLab, GitLab webhook URL: `http://[서비스도메인]:[Jenkins port]/project/[pipeline item명]` 체크

Enabled GitLab triggers

- ☒ Push Events 체크

Dashboard > peak > Configuration

Configure

- General**
- Advanced Project Options
- Pipeline

☒ Build when a change is pushed to GitLab. GitLab webhook URL: http/,

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

고급 ^

저장 Apply

Dashboard > peak > Configuration

Configure

- General**
- Advanced Project Options
- Pipeline

☐ Cancel pending merge request builds on update

Allowed branches

- ☒ Allow all branches to trigger this job ?
- ☐ Filter branches by name ?
- ☐ Filter branches by regex ?
- ☐ Filter merge request by label

Secret token ?

Generate

Clear

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

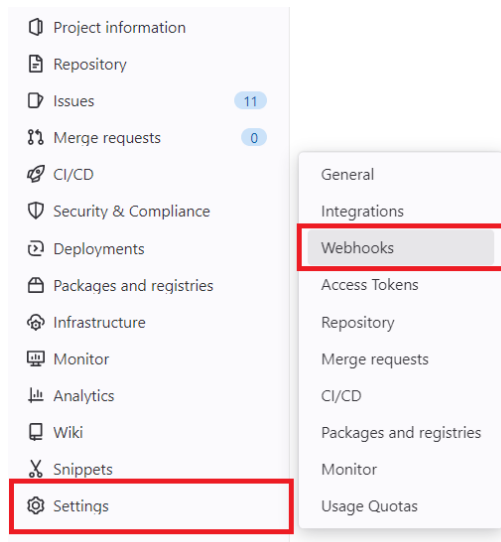
저장 Apply

고급 클릭

- Secret token > Generate 클릭 후 Copy 해두기

Gitlab (3)

16. Settings > Webhooks



Q 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

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.

☐ Tag push events
A new tag is pushed to the repository.

☐ Comments
A comment is added to an issue or merge request.

☐ Confidential comments
A comment is added to a confidential issue.

☐ Issues events
An issue is created, updated, closed, or reopened.

- URL : 15 에서 Build Triggers 설정했던 GitLab webhook URL: `http://[서비스도메인]:[Jenkins port]/project/[pipeline item명]` 입력
- Secret token : 15 에서 Generate해서 Copy 해둔 Secret token 입력
- Trigger : ☒ Push events 체크
 - master 입력 (events를 감지하여 webhook이 발생할 대상 branch 설정)

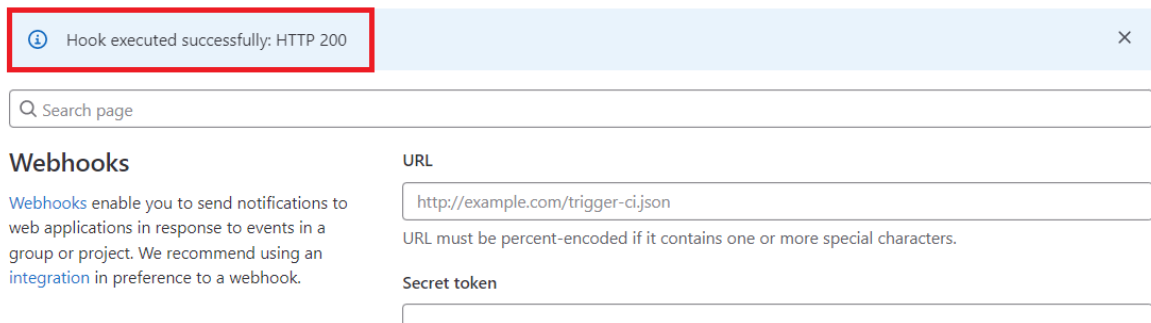
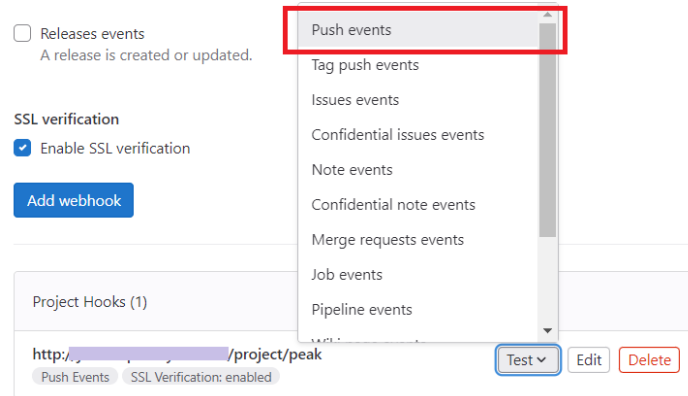
SSL verification

☒ Enable SSL verification

SSL verification

- ☒ Enable SSL verification 체크

⇒ Add webhook



하단 Project Hooks 목록에서 Test > Push events 클릭 후 최상단에 HTTP 200이 뜨면 성공

Jenkins (4)

17. 15 화면 (Dashboard > peak > Configuration) > Pipeline

Dashboard > peak > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://

Credentials ?

Add +

고급 ▾

Add Repository

저장 Apply

- Definition : Pipeline script from SCM 선택
- SCM : Git 선택

Repositories

- Repository URL : GitLab Repository URL 입력
- Credentials : 13 에서 설정했던 User Credential 선택

Dashboard > peak > Configuration

Configure

- General
- Advanced Project Options
- Pipeline**

Pipeline

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

Add Branch

Repository browser ?

(자동)

Additional Behaviours

Add +

Script Path ?

Jenkinsfile

☒ Lightweight checkout ?

저장 Apply

Branches to build

- Branch Specifier (blank for 'any') : ***/master 입력** (trigger 발생 시 build 대상이 되는 branch 설정)
- Script Path : Gitlab Repository 최상위 디렉토리 기준으로 Jenkinsfile의 경로 입력

⇒ 저장

! Jenkins react build 시 에러

Console Output

```
[Pipeline] sh
+ npm run build

> prac@0.1.0 build
> react-scripts build

Creating an optimized production build...

Treating warnings as errors because process.env.CI = true.
Most CI servers set it automatically.

Failed to compile.
```

원인

리액트에서는 환경변수 **CI** 값이 default **true**로 설정되어 있는데,

빌드시 WARN을 ERROR로 인식하는 경우라고 함

수동 배포는 잘 됐으나 젠킨스에서만 중지 됨

해결

Jenkins에서 환경 변수 설정 해줌

Dashboard > Jenkins 관리 > System >

Global properties

☐ Disable deferred wipeout on this node ?

☒ Environment variables

키-값 목록 ?

이름	값
CI	false

☐ Tool Locations

Pipeline Speed / Durability

Default Speed / Durability Level ?

Dashboard > Jenkins 관리 > System

Global properties

- ☒ Environment variables

카-값 목록

- 이름 : CI
- 값 : false

⇒ 저장

Jenkins, Gitlab 설정 후 Jenkins에서 지금 빌드

→ EC2 server console에서 ~/jenkins/workspace/[git repository명]/ 위치에 브랜치가 생성되었는지 확인

• Backend

```
cd ~/jenkins/workspace/peak/back/src/main/  
sudo mkdir resources  
sudo vim application.yml
```

⇒ application.yml 작성

• Frontend

```
cd ~/jenkins/workspace/peak/front/  
sudo vim .env
```

⇒ .env 작성

이후, 대상 브랜치에 push로 trigger가 실행되거나 Jenkins에서 지금 빌드를 누르면 배포 완료

IV. 외부 서비스

▼ 1. Kakao OAuth2.0

01) <https://developers.kakao.com/> 접속

02) 로그인 후, 내 애플리케이션 > 애플리케이션 추가하기



애플리케이션 추가하기

앱 아이콘	<div>이미지 업로드</div> <div>파일 선택</div> <div>JPG, GIF, PNG</div> <div>권장 사이즈 128px, 최대 250KB</div>
앱 이름	<input type="text" value="내 애플리케이션 이름"/>
사업자명	<input type="text" value="사업자 정보와 동일한 이름"/>

• 입력된 정보는 사용자가 카카오 로그인 할 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

- 앱 이름 : 임의 설정
- 사업자명 : 임의 설정 (팀명 입력하였음)
- ☒ 서비스 이용이 제한되는 카테고리, 금지된 내용, 금지된 행동 관련 운영 정책을 위반하지않는 앱입니다. 체크

03) 전체 애플리케이션 목록 > 앱 설정 > 요약정보

앱 키

네이티브 앱 키

REST API 키

JavaScript 키

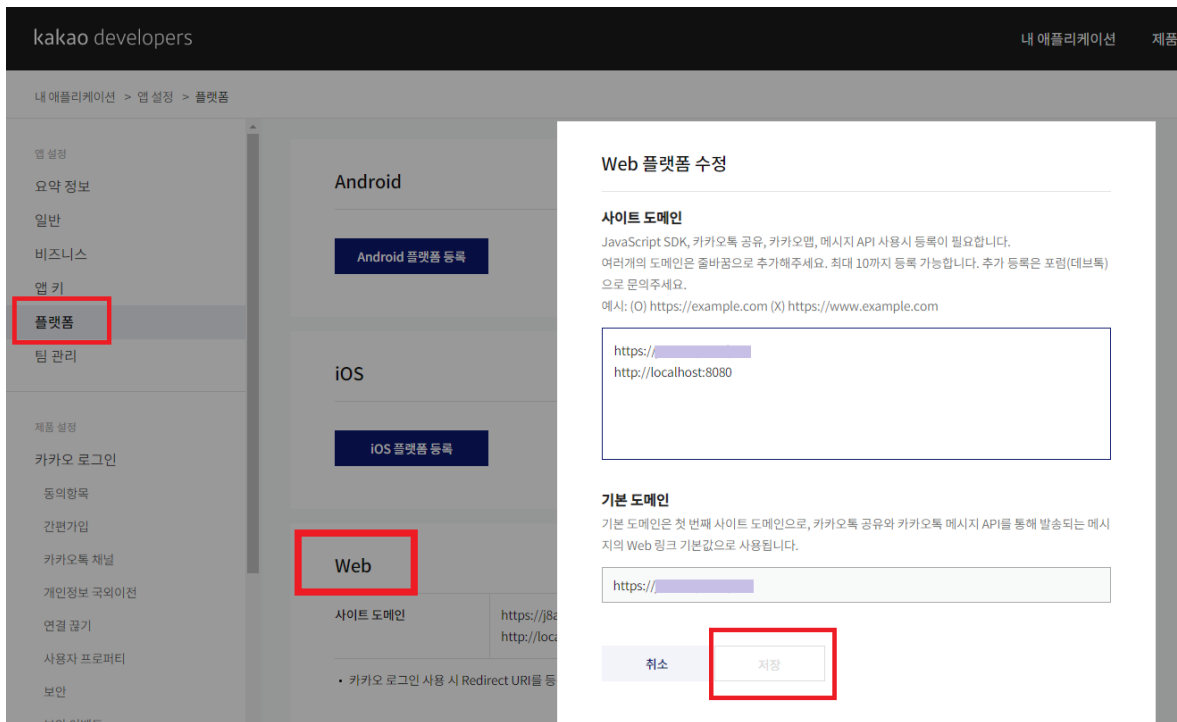
Admin 키

REST API 를 spring boot app의 src > main > resources > application.yml 의

```

spring:
  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: 에 입력
          
```

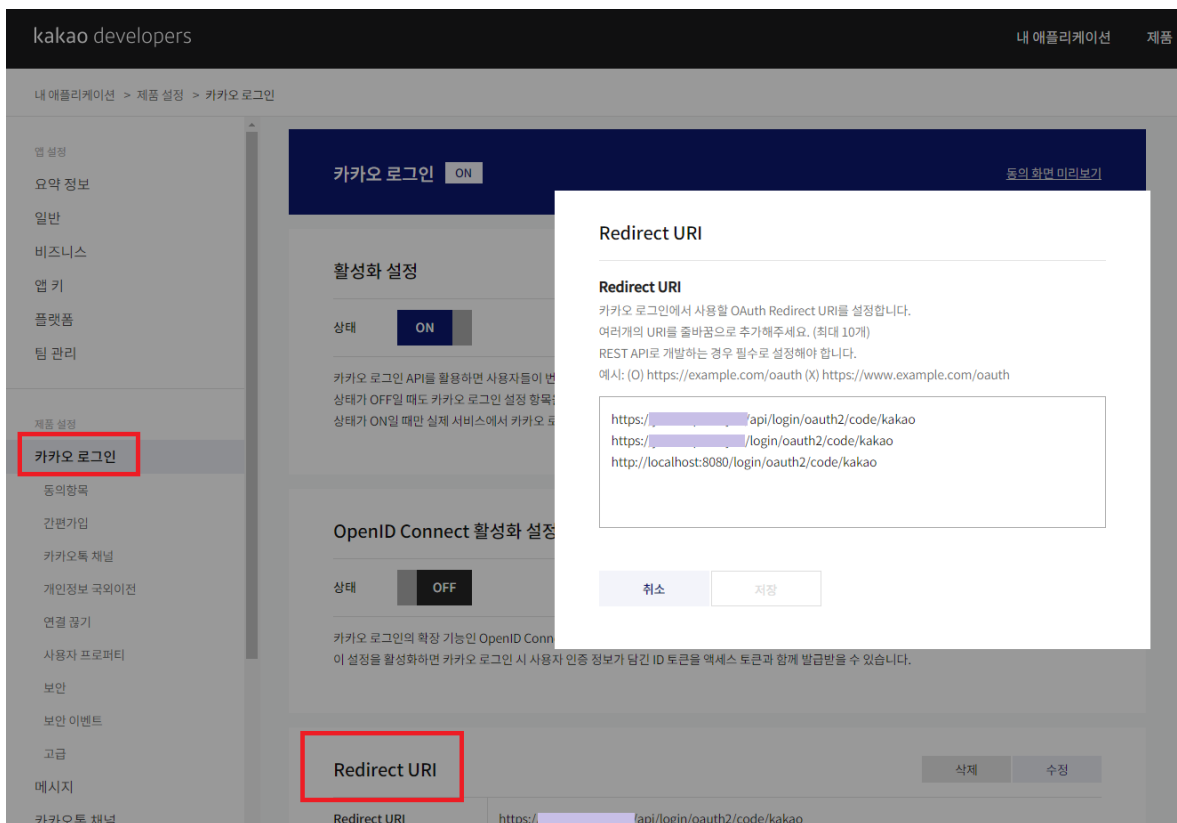
04) 내 애플리케이션 > 앱 설정 > 플랫폼 > Web



사이트 도메인 등록

- 서비스 도메인, 로컬 등등 kakao OAuth를 사용할 도메인 등록

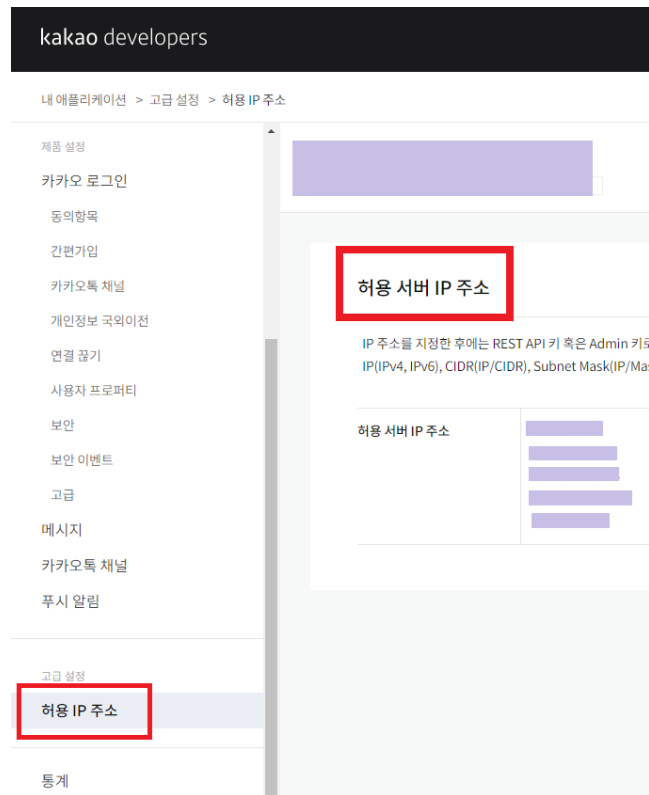
05) 내 애플리케이션 > 제품 설정 > 카카오 로그인



Redirect URI 등록

- [https://\[서비스도메인\]/login/oauth2/code/kakao](https://[서비스도메인]/login/oauth2/code/kakao) : 해당 uri가 Spring Security의 기본 format

06) 내 애플리케이션 > 고급 설정 > 허용 IP 주소



- 허용 서버 IP 주소 등록

07) 내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목

kakao developers

내 애플리케이션

내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

간편가입

카카오톡 채널

개인정보 국외이전

연결 끊기

사용자 프로퍼티

보안

보안 이벤트

고급

메시지

카카오톡 채널

푸시 알림

카카오 로그인 ON

동의 화면 미리보기

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	● 사용 안함 <div>설정</div>
프로필 사진	profile_image	● 사용 안함 <div>설정</div>
카카오계정(이메일)	account_email	● 필수 동의 [수집] <div>설정</div>
이름	name	○ 권한 없음
성별	gender	● 사용 안함 <div>설정</div>

- 필요한 유저 정보에 대해 동의 항목 설정 추가
 - 카카오계정(이메일) 추가 시 비즈앱 등록 필수

07-1) 내 애플리케이션 > 앱 설정 > 비즈니스

kakao developers

내 애플리케이션 > 앱 설정 > 비즈니스

앱 설정

요약 정보

일반

비즈니스

앱 키

플랫폼

팀 관리

제품 설정

카카오 로그인

동의항목

APP

비즈 앱 정보

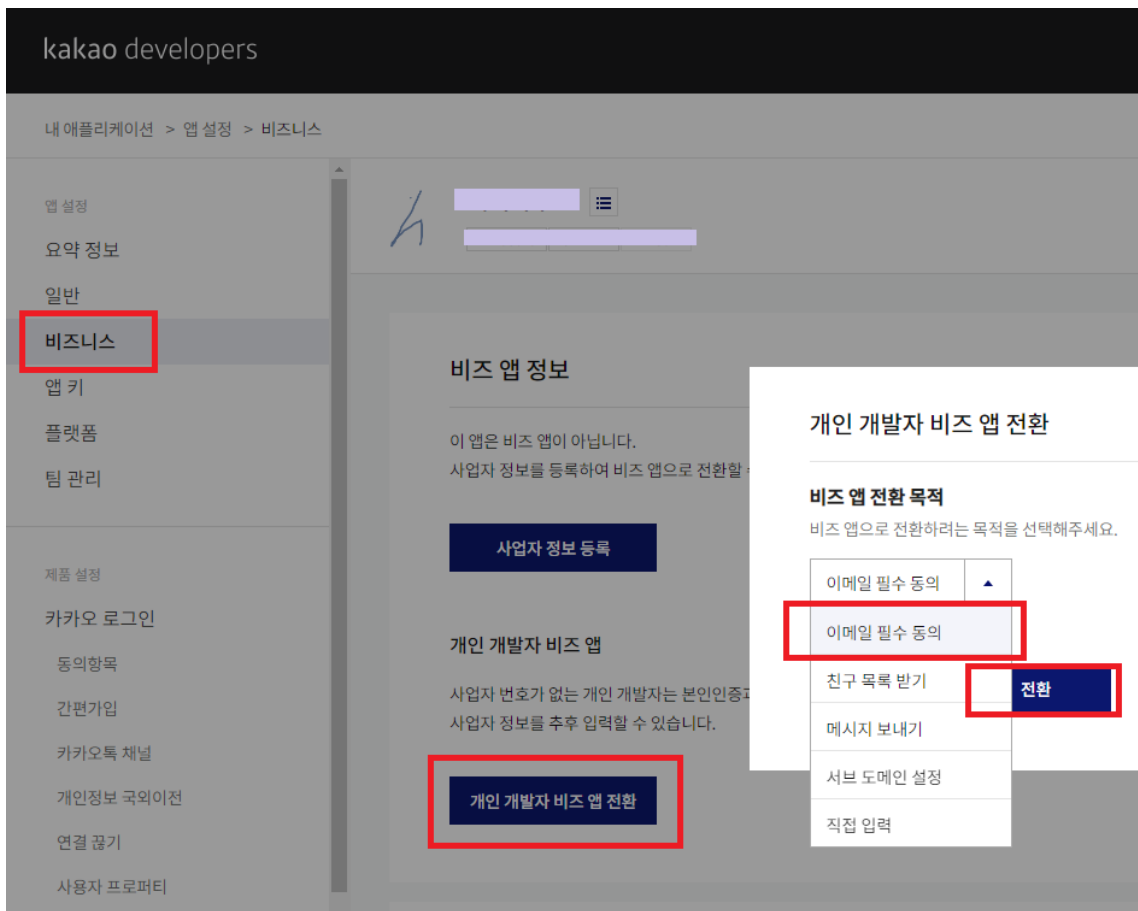
이 앱은 비즈 앱이 아닙니다.

사업자 정보를 등록하여 비즈 앱으로 전환할 수 있
사업자 번호가 없는 개인 개발자는 본인인증과 카:

비즈 앱 전환을 위해 앱 아이콘 등록이 필요합니다

앱 아이콘 등록

- 비즈 앱 등록 시 앱 아이콘 등록 필수



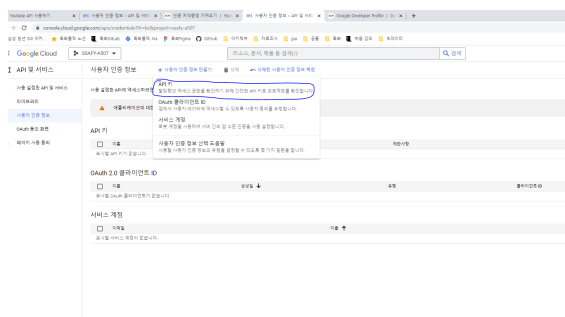
- 개인 개발자 비즈 앱 전환 > 비즈 앱 전환 목적 선택

▼ 2. YouTube Data API

API Key 발급

- 개발자 콘솔

<https://console.cloud.google.com/apis/credentials?hl=ko&project=ssafy18-06>



프로젝트 생성 → 사용자 인증 정보 만들기 → API key 생성

의존성 추가

- Gradle dependency 에 `google-api-services-youtube:v3-rev20230123-2.0.0` , `google-http-client-jackson2` 추가

사용해보기

Search | YouTube Data API | Google Developers
<https://developers.google.com/youtube/v3/docs/search?hl=ko>

• Request

channelId	string <code>channelId</code> 매개변수는 API 응답이 채널에서 만든 특정 리소스만 포함해야 한다는 것을 나타냅니다.
maxResults	unsigned integer <code>maxResults</code> 매개변수는 결과 집합에 반환해야 하는 최대 항목 수를 지정합니다. 허용값은 0 이상 50 이하입니다. 기본값은 5입니다.
order	string-order 매개변수는 API 응답에서 리소스를 지시하는 데 사용할 메소드를 지정합니다. 기본값은 SEARCH_SORT_RELEVANCE입니다. 허용값은 다음과 같습니다.- <code>viewCount</code> - 리소스를 조회수가 높은 항목부터 정렬합니다.
q	string <code>q</code> 매개변수는 검색할 검색어를 지정합니다.

• Response

<code>snippet.title</code>	string 검색결과와 제목입니다.
<code>snippet.thumbnails</code>	object 검색결과에 관련된 미리보기 이미지 맵입니다. 맵의 각 개체에 대해, 키는 미리보기 이미지의 이름이고 값은 미리보기 이미지에 대한 기타 정보를 포함하는 개체입니다.
<code>snippet.thumbnails.(key).url</code>	string 이미지의 URL입니다. object 유효한 키 값은 다음과 같습니다. • high - 미리보기 이미지의 고해상도 버전입니다. 동영상 또는 동영상을 참조하는 리소스의 경우 이 이미지는 480x360픽셀입니다. 채널의 경우 이 이미지는 800x800픽셀입니다.

Controller, Service 코드 작성

<https://developers.google.com/youtube/v3/docs/search/list?hl=ko#예>

- 예제 코드에서 원하는 정보에 맞춰 변경하여 작성

3. Google Elastic Search