



Porting Manual : peak

특화 A507 하아코손

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
 - Swagger
 - JWT 0.11.5
- Frontend
 - Visual Studio Code
 - Node.js
 - React
 - Redux
 - TypeScript
 - Apache ECharts
- Database
 - MongoDB
 - Redis

2. 디렉토리 구조

▼ Cluster server

```
├─ 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
├─ docker-compose.yml
├─ idol_images
│  ├── BTS.webp
│  ├── 세븐틴.webp
│  ├── ...
│  ├── ...
│  └── NewJeans.webp
├─ jenkins
│  ├── caches
│  ├── com.cloudbees.hudson.plugins.folder.config.AbstractFolderConfiguration.xml
│  ├── com.dabsquared.gitlabjenkins.GitLabPushTrigger.xml
│  ├── com.dabsquared.gitlabjenkins.connection.GitLabConnectionConfig.xml
│  ├── config.xml
│  ├── copy_reference_file.log
│  ├── credentials.xml
│  ├── fingerprints
│  ├── github-plugin-configuration.xml
│  ├── hudson.model.UpdateCenter.xml
│  ├── hudson.plugins.build_timeout.global.GlobalTimeoutConfiguration.xml
│  ├── hudson.plugins.build_timeout.operations.BuildStepOperation.xml
│  ├── hudson.plugins.emailxt.ExtendedEmailPublisher.xml
│  ├── hudson.plugins.git.GitSCM.xml
│  ├── hudson.plugins.git.GitTool.xml
│  ├── hudson.plugins.gradle.Gradle.xml
│  ├── hudson.plugins.gradle.enriched.EnrichedSummaryConfig.xml
│  ├── hudson.plugins.gradle.injection.InjectionConfig.xml
│  ├── hudson.plugins.timestamp.TimestamperConfig.xml
│  ├── hudson.tasks.Ant.xml
│  ├── hudson.tasks.Mailer.xml
│  ├── hudson.tasks.Maven.xml
│  ├── hudson.tasks.Shell.xml
│  ├── hudson.triggers.SCMTrigger.xml
│  ├── identity.key.enc
│  ├── io.jenkins.plugins.junit.storage.JunitTestResultStorageConfiguration.xml
│  ├── jenkins.fingerprints.GlobalFingerprintConfiguration.xml
│  ├── jenkins.install.InstallUtil.lastExecVersion
│  ├── jenkins.install.UpgradeWizard.state
│  ├── jenkins.model.ArtifactManagerConfiguration.xml
│  ├── jenkins.model.GlobalBuildDiscarderConfiguration.xml
│  ├── jenkins.model.JenkinsLocationConfiguration.xml
│  ├── jenkins.mvn.GlobalMavenConfig.xml
│  ├── jenkins.plugins.nodejs.tools.NodeJSInstallation.xml
│  ├── jenkins.security.ResourceDomainConfiguration.xml
│  ├── jenkins.tasks.filters.EnvVarsFilterGlobalConfiguration.xml
│  ├── jenkins.telemetry.Correlator.xml
│  ├── jobs
│  ├── logs
│  ├── nodeMonitors.xml
│  ├── nodes
│  ├── org.jenkinsci.plugins.gitclient.JGitApacheTool.xml
│  ├── org.jenkinsci.plugins.gitclient.JGitTool.xml
│  ├── org.jenkinsci.plugins.github_branch_source.GitHubConfiguration.xml
│  ├── org.jenkinsci.plugins.workflow.flow.FlowExecutionList.xml
│  ├── org.jenkinsci.plugins.workflow.flow.GlobalDefaultFlowDurabilityLevel.xml
│  ├── org.jenkinsci.plugins.workflow.libs.GlobalLibraries.xml
│  ├── plugins
│  ├── queue.xml
│  ├── queue.xml.bak
│  ├── scriptApproval.xml
│  ├── secret.key
│  ├── secret.key.not-so-secret
│  ├── secrets
│  ├── tools
│  ├── updates
│  ├── userContent
│  ├── users
│  └── war
```

```

|   └─ workspace
├─ miniconda3
|   └─ LICENSE.txt
|   └─ bin
|   └─ compiler_compat
|   └─ conda-meta
|   └─ condabin
|   └─ envs
|   └─ etc
|   └─ include
|   └─ lib
|   └─ man
|   └─ pkgs
|   └─ share
|   └─ shell
|   └─ ssl
|   └─ x86_64-conda-linux-gnu
|   └─ x86_64-conda_cos7-linux-gnu
├─ mongo
|   └─ docker-compose.yml
├─ mongodb
├─ nginx
|   └─ conf.d
|   └─ var
├─ 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

```

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 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
```

Jenkins

1. Dockerfile 저장

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

2. Jenkins, Gitlab 설정 후 Jenkins에서 **지금 빌드**

→ EC2 server console에서 ~/jenkins/workspace/[git repository명]/ 경로에 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

IV. 외부 서비스

1. Kakao OAuth2.0

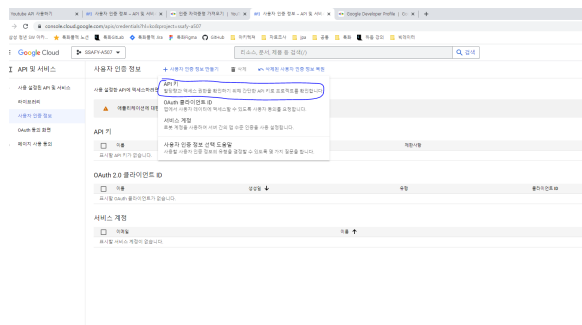
- API key 발급
- Redirect url 등록
- 허용 IP 등록
- 정보 동의 요청이 필요한 유저 정보에 대해 추가

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

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

- 예제 코드에서 원하는 정보에 맞춰 변경하여 작성

3. Google Elastic Search