



포팅 메뉴얼

1.1 버전 정리

[Frontend](#)

[Backend](#)

[Crawling](#)

[Hadoop](#)

[Database](#)

[Infra](#)

[IDE](#)

1.2 환경 변수

[Frontend](#)

[Backend](#)

1.3 배포 특이사항

1.3.1 방화벽 설정

[방화벽 상태 조회](#)

[방화벽 ssh 포트 오픈 후 활성화](#)

1.3.2 Docker 초기 설치

1.3.3 Nginx 설치

1.3.4 Docker sudo 권한 부여

1.3.5 Docker Mysql 설정

[mysql workbench 세팅](#)

1.3.6 Docker MongoDB 설정

[Studio 3T 세팅](#)

1.3.7 Nginx reverse proxy 설정

1.3.8 Frontend (nginx/react) Docker 설정

1.3.9 Backend (Spring boot) Docker 설정

1.3.10 Backend (Django) Docker 설정

2. 외부 서비스

2.1 카카오 로그인

[1. 앱 등록 후 앱 키 발급](#)

[2. \[내 애플리케이션\] > \[카카오 로그인\] 카카오 로그인 활성화 ON, Redirect URL 등록](#)

[3. \[내 애플리케이션\] > \[카카오 로그인\] > \[동의항목\]에서 동의 항목 설정](#)

2.2 AWS S3

[1. AWS 계정 생성](#)

[2. 버킷 생성](#)

[3. REST API로 데이터 업로드](#)

2.3 Youtube API

[1. Google API 콘솔에 액세스하고 API 키를 요청하며 애플리케이션을 등록하려면 Google 계정이 필요합니다.](#)

[2. API 요청을 제출할 수 있도록 Google에 애플리케이션을 등록합니다.](#)

[3. 애플리케이션을 등록한 후 다음과 같이 애플리케이션에서 사용할 서비스 중 하나로 YouTube Data API를 선택합니다.](#)

1.1 버전 정리

Frontend

- Node.js **18.13.0 (LTS)**
- React **18.2.0**
 - Recoil **0.7.6**
- typescript **4.9.5**
- react-dom **18.2.0**
- axios **1.2.6**
- styled-components **5.3.6** (주의 최신버전아님!)

Backend

- java 1.8
- spring 2.7.9
- jpa 2.2
- querydsl 5.0.0
- gradle 7.6.1
- django 4.2

Crawling

- python 3.8.10
- pymysql 1.0.2
- pandas 1.5.3
- selenium 4.8.2

Hadoop

- hadoop 3.3.1
- spark 3.0.16

Database

- mysql 8.0.30
- mongodb 6.0.3
- AWS S3 1.12.281

Infra

- ubuntu 20.04 LTS
- nginx 1.18.0 (Ubuntu)
- jenkins 2.387.1
- docker 23.0.2
- sonarqube 4.8.0.2856

IDE

- intellij 2022.3.1
- pycharm 2022.3.1
- vscode 1.77.0

1.2 환경 변수

Frontend

```
REACT_APP_API
REACT_APP_KAKAO_API
REACT_APP_KAKAO_CLIENT_ID
REACT_APP_KAKAO_CLIENT_SECRET
REACT_APP_KAKAO_REDIRECT_URI
REACT_APP_YOUTUBE_API_KEY
```

Backend

```

# MySQL
spring:
  datasource:
    driver-class-name:
    url:
    username:
    password:
  data:
    mongodb:
      host:
      port:
      username:
      password:
      authentication-database:
      database:
# Social Login
security:
  oauth2:
    client:
      registration:
        kakao:
          client-id:
          client-secret:
          client-name:
          authorization-grant-type:
          redirect-uri:
          client-authentication-method:
          scope:

      provider:
        kakao:
          authorization-uri:
          token-uri:
          user-info-uri:
          user-name-attribute:

# SMTP
mail:
  host:
  port:
  username:
  password:
  properties:
    mail:
      smtp:
        auth:
        starttls:
        enable:

# s3 file size bigger setting
servlet:
  multipart:
    max-file-size:
    max-request-size:

# jwt secret key
jwt:
  secret:
  # auth : | oauth2 :
app:
  auth:
    token-secret:
    token-expiry:
    refresh-token-expiry:
  oauth2:
    authorized-redirect-uris:

# S3 Setting
cloud:
  aws:
    credentials:
      accessKey: # AWS IAM AccessKey 적기
      secretKey: # AWS IAM SecretKey 적기
    s3:
      bucket: # 버킷 이름
      dir: # s3 디렉토리 이름
      region:
      static:
    stack:
      auto:

```

1.3 배포 특이사항

1.3.1 방화벽 설정

방화벽 상태 조회

```
sudo ufw status verbose
```

방화벽 ssh 포트 오픈 후 활성화

```
sudo ufw allow ssh
```

```
sudo ufw enable
```

1.3.2 Docker 초기 설치

```
# Uninstall old version
sudo apt-get remove docker docker-engine docker.io containerd runc

# Install using the repository

## Update the apt package index and install packages to allow apt to use a repository over HTTPS:
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

## Add Docker's official GPG key:
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

## Use the following command to set up the repository:
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

# Install Docker Engine

## Update the apt package index
sudo apt-get update

## Install Docker Engine, containerd, and Docker Compose. (latest version)
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```

1.3.3 Nginx 설치

```
sudo apt update
sudo apt install nginx

# 서버에서 사용할 수 있는 모든 응용프로그램 프로파일 나열
sudo ufw app list
# sudo ufw allow 'Nginx HTTP'
# sudo ufw allow 'Nginx HTTPS'
sudo ufw allow 'Nginx FULL'
```

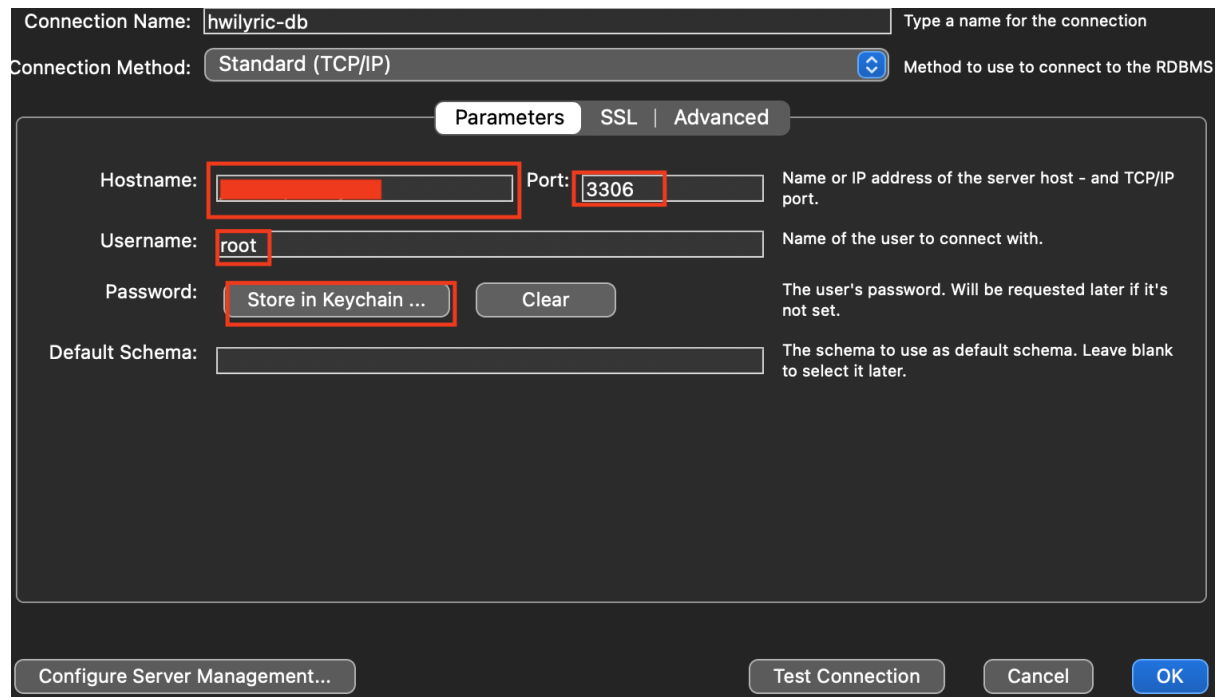
1.3.4 Docker sudo 권한 부여

```
# docker 그룹에 사용자 추가
sudo groupadd docker
sudo usermod -aG docker {user_id}
# docker 재시작
sudo service docker restart
# 시스템 재시작
sudo systemctl reboot
```

1.3.5 Docker Mysql 설정

```
sudo docker pull mysql:8.0.30
docker run --name mysql-container -v /home/ubuntu/mysql:/data/db -e MYSQL_ROOT_PASSWORD={YOUR_PASSWORD} -d -p 3306:3306 mysql:8.0.30
docker exec -it mysql-container bash
mysql -u root -p
```

mysql workbench 세팅



The screenshot shows the MySQL Workbench Connection Wizard. The 'Connection Name' is 'hwilyric-db'. The 'Connection Method' is 'Standard (TCP/IP)'. The 'Parameters' tab is selected. The 'Hostname' field is empty, and the 'Port' is '3306'. The 'Username' is 'root'. The 'Password' field has a 'Store in Keychain ...' button. The 'Default Schema' is empty. The 'Test Connection' button is highlighted.

Connection Name: Type a name for the connection

Connection Method: Method to use to connect to the RDBMS

Parameters | SSL | Advanced

Hostname: Port: Name or IP address of the server host - and TCP/IP port.

Username: Name of the user to connect with.

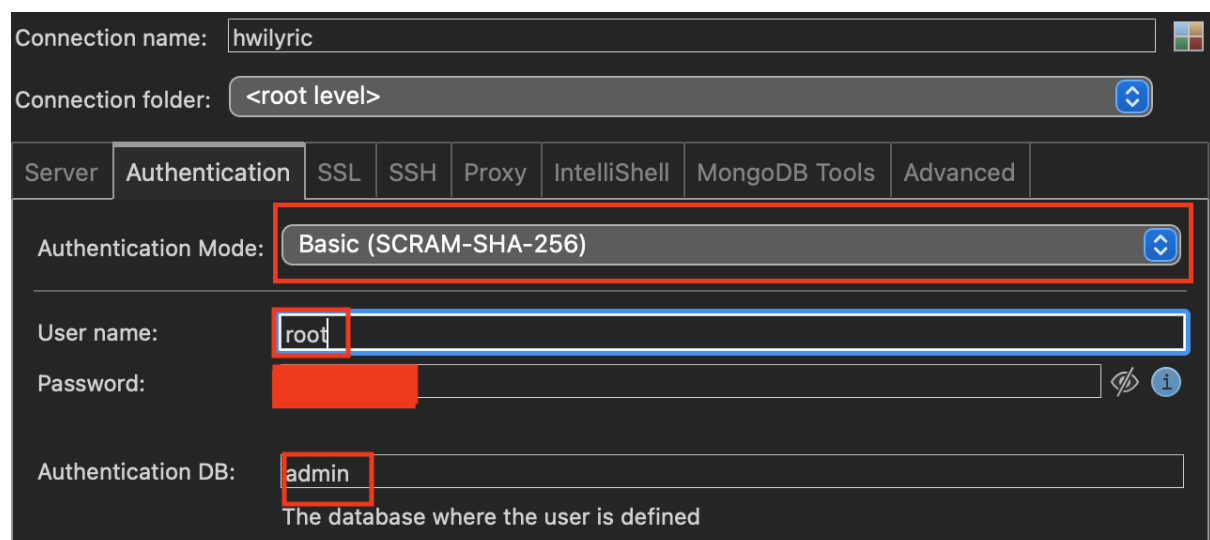
Password: The user's password. Will be requested later if it's not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

1.3.6 Docker MongoDB 설정

```
docker pull mongo:6.0.3
docker run --name mongodb-container -v /home/ubuntu/mongodb:/data/db -d -p 27017:27017 -e MONGO_INITDB_ROOT_USERNAME=root -e MONGO_INITDB_ROOT_PASSWORD=12345678
# mongosh 접속
docker exec -it mongodb-container mongosh -u root -p {YOUR_PASSWORD}
```

Studio 3T 세팅



The screenshot shows the Studio 3T Connection Wizard. The 'Connection name' is 'hwilyric'. The 'Connection folder' is '<root level>'. The 'Authentication Mode' is 'Basic (SCRAM-SHA-256)'. The 'User name' is 'root'. The 'Password' field is empty. The 'Authentication DB' is 'admin'. The 'Test Connection' button is highlighted.

Connection name:

Connection folder:

Server | Authentication | SSL | SSH | Proxy | IntelliShell | MongoDB Tools | Advanced

Authentication Mode:

User name:

Password:

Authentication DB: The database where the user is defined

Connection name:

Connection folder:

Server Authentication SSL SSH Proxy IntelliShell MongoDB Tools Advanced

Connection Type:

Server:

Port:

admin 권한 사용 시 unauthorized(13) 에러가 뜨는 이슈가 있었다. `vim /etc/mongod.conf.orig` 를 통해 security 주석을 해제하고 하위에 authentication: 를 추가해주었다. 여전히 에러는 해결되지 않았지만 admin DB가 아닌 다른 DB에서 작업하는 것은 성공했다.

1.3.7 Nginx reverse proxy 설정

```
cd /etc/nginx/sites-available
vi configure

---
limit_req_zone $binary_remote_addr zone=ddos_req:10m rate=30r/s;

server {
    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/j8b107.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/j8b107.p.ssafy.io/privkey.pem;

    location / {
        add_header 'Access-Control-Allow-Origin' '*';
        add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
        limit_req zone=ddos_req burst=5;
        proxy_pass http://localhost:3000;
    }

    location /api {
        add_header 'Access-Control-Allow-Origin' '*';
        add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
        limit_req zone=ddos_req burst=5;
        proxy_pass http://localhost:8080/api;
    }

    location /recommend {
        add_header 'Access-Control-Allow-Origin' '*';
        add_header Access-Control-Allow-Methods 'GET, POST, OPTIONS';
        limit_req zone=ddos_req burst=5;
        proxy_read_timeout 180s;
        proxy_send_timeout 180s;
        proxy_pass http://localhost:8080/recommend;
    }
}

server {
    if ($host = j8b107.p.ssafy.io) {
        return 301 https://$host$request_uri;
    }

    listen 80;
    server_name j8b107.p.ssafy.io;
    return 404;
}
---
```

```
sudo ln -s /etc/nginx/sites-available/configure /etc/nginx/sites-enabled/configure # 파일 링크
sudo nginx -t # ok 시 성공
sudo systemctl restart nginx
```

1.3.8 Frontend (nginx/react) Docker 설정

```

# 1. Home Directory로 이동
cd /home/ubuntu

# 2. Git Clone
git clone {깃랩주소}

# 3. 프론트엔드 root 폴더로 이동
cd {프로젝트/프론트엔드폴더}

# 4. nginx config 파일 생성
vi nginx.conf
---
server {
    listen 80;
    location / {
        root /app/build;
        index index.html;
        try_files $uri $uri/ /index.html;
    }
}
---
# 5. Dockerfile 생성
vi Dockerfile
---
FROM nginx
RUN mkdir /app
WORKDIR /app
RUN mkdir ./build
ADD ./build ./build
RUN rm /etc/nginx/conf.d/default.conf
COPY ./nginx.conf /etc/nginx/conf.d
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
---

# nginx 컨테이너 접속
# 6. nvm(Node Version Manager) + node 다운로드
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
export NVM_DIR="$([ -z "${XDG_CONFIG_HOME-}" ] && printf %s "${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
nvm install 18.13.0 # or 16.3.0, 12.22.1, etc
nvm run node --18.13.0

# 7. 빌드 파일 생성
npm run build

# 8. 도커 이미지 생성 및 컨테이너 생성
docker build -t {이미지 이름} .
docker rm -f {기존 컨테이너 이름}
docker run --name {이미지 이름} -d -p 3000:80 {컨테이너 이름}

```

1.3.9 Backend (Spring boot) Docker 설정

```

# sdk 설치 후 gradle 설치(최초 한 번)
sudo apt install unzip
sudo apt install zip
curl -s "https://get.sdkman.io" | bash
source "$HOME/.sdkman/bin/sdkman-init.sh"
sdk version # 버전이 나오면 성공
sdk install gradle 7.6.1

# Java 설치 및 환경 변수 설정 (최초 한 번)
sudo apt-get install openjdk-8-jdk
javac -version # 버전이 나오면 성공
which javac
readlink -f /usr/bin/javac
sudo vi /etc/profile
---
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/bin/javac
---
source /etc/profile # 환경 변수 적용
echo $JAVA_HOME # 잘 나오는지 확인

# 1. 위치 이동
cd /home/ubuntu/프로젝트폴더/백엔드폴더

# 2. Dockerfile 생성
vim Dockerfile
---
FROM openjdk:8-jdk-alpine
RUN apk update \
&& apk upgrade \

```

```

&& apk add --no-cache bash \
&& apk add --no-cache --virtual=build-dependencies unzip \
&& apk add --no-cache curl

RUN apk add --no-cache python3 \
&& python3 -m ensurepip \
&& pip3 install --upgrade pip setuptools \
&& rm -r /usr/lib/python*/ensurepip && \
if [ ! -e /usr/bin/pip ]; then ln -s pip3 /usr/bin/pip ; fi && \
if [ ! -e /usr/bin/python ]; then ln -sf /usr/bin/python3 /usr/bin/python; fi && \
rm -r /root/.cache

VOLUME /tmp
ARG JAR_FILE=build/libs/hwilyric-0.0.1-SNAPSHOT.jar
COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
---

# 3. gradle 빌드 실행
gradle clean build test

# 4. 도커 이미지 생성
docker build -t spring-image .
docker rm -f hl-backend
docker image prune

# 5. 도커 컨테이너 생성
docker run --name hl-backend -d -p 8080:8080 spring-image

```

1.3.10 Backend (Django) Docker 설정

```

# 1. 위치 이동
cd /home/ubuntu/프로젝트폴더/백엔드장고폴더

# 2. Dockerfile 생성
FROM python:3.8
ENV PYTHONUNBUFFERED 1
RUN apt-get -y update && apt-get -y install vim && apt-get clean && apt-get -y install libgl1-mesa-glx && apt-get install -y python3-pip
RUN apt-get install default-jdk -y
RUN mkdir /project
ADD . /project
WORKDIR /project
RUN pip install --upgrade pip
RUN pip install -r requirements.txt
EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

# 3. 도커 이미지 생성
docker rmi django-image
docker build -t django-image .
docker rm -f django-image

# 4. 도커 컨테이너 생성
docker run --name hl-django -d -p 8000:8000 django-image

```

2. 외부 서비스

2.1 카카오 로그인

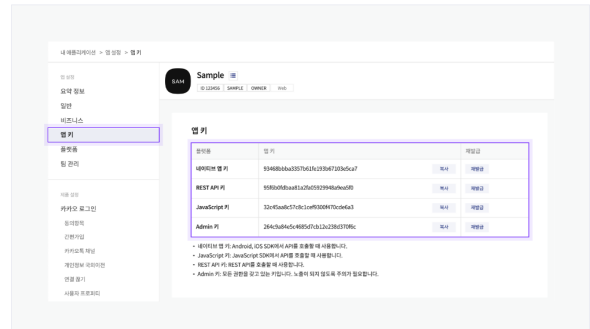
사용자의 회원가입 절차 간소화 및 보안 향상을 위해 카카오 로그인 기능을 도입했습니다.

<https://developers.kakao.com/docs/latest/ko/getting-started/app>

1. 앱 등록 후 앱 키 발급

2. [내 애플리케이션] > [카카오 로그인] 카카오 로그인 활성화 ON, Redirect URL 등록

3. [내 애플리케이션] > [카카오 로그인] > [동의항목]에서 동의 항목 설정



2.2 AWS S3

사용자의 데이터를 안전하게 저장 및 관리하기 위해 AWS S3를 사용했습니다.

https://docs.aws.amazon.com/ko_kr/s3/index.html

1. AWS 계정 생성

AWS에 가입

루트 사용자 이메일 주소
계정 복구 및 일부 관리 기능에 사용

AWS 계정 이름
계정의 이름을 선택합니다. 이름은 가입 후 계정 설정에서 변경할 수 있습니다.

이메일 주소 확인

2. 버킷 생성

1. AWS Management Console에 로그인한 후 <https://console.aws.amazon.com/s3/>에서 Amazon S3 콘솔을 엽니다.
2. 버킷 만들기를 선택합니다.
버킷 만들기 마법사가 열립니다.
3. 버킷 이름에 버킷의 DNS 호환 이름을 입력합니다.
4. [리전(Region)]에서 버킷이 속할 AWS 리전을 선택합니다.
5. 객체 소유권(Object Ownership)에서 ACL을 사용 중지 또는 사용 설정합니다.
6. Bucket settings for Block Public Access(퍼블릭 액세스 차단을 위한 버킷 설정)에서 버킷에 적용할 퍼블릭 액세스 차단 설정을 선택합니다.
7. 버킷 만들기를 선택합니다.

버킷 만들기 Info
 버킷은 S3에 저장되는 데이터의 컨테이너입니다. 자세히 알아보기 [🔗](#)

일반 구성

버킷 이름

버킷 이름은 전역에서 고유해야 하며 공백 또는 대문자를 포함할 수 없습니다. 버킷 이름 지정 규칙 참조 [🔗](#)

AWS 리전

기존 버킷에서 설정 복사 - 선택 사항
 다음 구성의 버킷 설정만 복사됩니다.

3. REST API로 데이터 업로드

2.3 Youtube API

작사에 필요한 음악 플레이어 지원을 위해 Youtube API를 사용했습니다.

<https://developers.google.com/youtube/v3/getting-started?hl=ko>

1. Google API 콘솔에 액세스하고 API 키를 요청하며 애플리케이션을 등록하려면 Google 계정이 필요합니다.

2. API 요청을 제출할 수 있도록 Google에 애플리케이션을 등록합니다.

3. 애플리케이션을 등록한 후 다음과 같이 애플리케이션에서 사용할 서비스 중 하나로 YouTube Data API를 선택합니다.

1. APIs Console로 이동하고 앞에서 등록한 프로젝트를 선택합니다.
2. **Services** 창을 클릭합니다.
3. API 목록에서 **YouTube Data API**를 찾아 상태를 **ON**으로 변경합니다.