





Churros 포팅매뉴얼

👤 생성자	 sohee yoon
🕒 생성 일시	@2023년 4월 3일 오전 9:27
👤 최종 편집자	 sohee yoon
🕒 최종 편집 일시	@2023년 4월 7일 오전 9:59
≡ 태그	

삼성 SW 청년 아카데미 서울캠퍼스 8기

특화프로젝트 (7주, 2023/02/20 - 2023/04/07)

0. 팀 구성

1. 프로젝트 기술 스택

A. 공통

B. FE

C. BE

i. WAS (Web Application Server)

ii. CRS (Contents Recommendation Server)

iii. 공통

D. DATA

i. 크롤링

ii. 빅데이터 추천 모델

2. EC2 세팅

A. Docker 세팅

i. Docker 설치

ii. Docker compose 설치

iii. 도커 네트워크 생성

B. HTTPS 설정

i. Certbot 설치

ii. Nginx 설치 및 설정

iii. Certbot을 nginx에 적용 후 재시작

3. 프로퍼티 정의

A. Nginx Default 값 세팅

i. nginx.conf

ii. reverse proxy

B. Git Ignore 파일 설정

4. Jenkins 세팅

A. Jenkins 구성

i. 젠킨스 이미지 다운로드 및 실행

ii. Gitlab, Docker 플러그인 설정

iii. jenkins에 docker 접근 설정

5. 데이터베이스 세팅

A. Mariadb 설정

i. Dockerfile 작성

ii. mariadb.cnf 작성

iii. Dockerfile 빌드 및 실행

B. Monbodb 설정

i. Dockerfile작성

ii. mongod.conf 설정

iii. Dockerfile 빌드 및 실행

C. Elasticsearch 설정

i. Dockerfile 작성

ii. Dockerfile 빌드 및 실행

D. Monstache 설정

i. Dockerfile 작성

- ii. [conf/config.toml](#) 작성
 - iii. [Dockerfile](#) 빌드 및 실행
- 6. [DB 계정](#)
 - A. [mariaDB](#)
 - B. [MongoDB](#)
 - i. [admin 계정](#)
 - ii. [user 계정](#)
 - C. [ElasticSearch](#)
- 7. [빌드 방법](#)
 - A. [백엔드 빌드 방법](#)
 - i. [데이터베이스 세팅](#)
 - ii. [WAS Dockerfile](#) 작성
 - iii. [CRS Dockerfile](#) 작성
 - iv. [최상위 폴더에서 docker-compose up](#) 실행
 - B. [프론트엔드 빌드 방법](#)
- 8. [외부 서비스](#)
 - A. [카카오 로그인](#)

0. 팀 구성

- 담당 컨설턴트 : 한기철
- 강승곤(팀장), 서주광, 윤상빈, 윤소희, 이안채, 홍영민

1. 프로젝트 기술 스택

A. 공통

이슈 관리 : Jira

형상 관리 : Git

협동 툴 : Mattermost, Webex, Notion

Server : AWS EC2 [Ubuntu 20.04 LTS](#)

B. FE

기술 스택(버전) :

사용 툴 :

C. BE

i. WAS (Web Application Server)

기술 스택(버전) : [openjdk 11](#), [Spring boot 2.7.9](#), [OAuth2](#),

사용 툴 :

ii. CRS (Contents Recommendation Server)

기술 스택(버전) : [Python3 3.9.2](#), [fastapi 0.95.0](#), [uvicorn 0.21.1](#), [SQLAlchemy 2.0.8](#), [PyMySQL 1.0.3](#), [numpy 1.24.2](#), [gensim 4.3.1](#), [pytest 7.1.2](#)

사용 툴 : [PyCharm 2022.3.2](#)

iii. 공통

기술 스택(버전) : [Nginx 1.18.0](#), [Jenkins](#), [Docker 23.0.1](#)

사용 툴 : [Mobaxterm](#)

D. DATA

i. 크롤링

기술 스택(버전) : Python3 3.9.2, MongoDB 4.2, Crontab, Shell Script

사용 툴 : PyCharm 2022.3.2, Mobaxterm

ii. 빅데이터 추천 모델

기술 스택(버전) : Python 3.9, Numpy 1.24, Pandas 1.5, konlpy 0.6, gensim 4.3.1, seaborn 0.12, matplotlib 3.7, pyLDAvis 3.4, MongoDB 4.2, openjdk 11

사용 툴 : JupyterNotebook 5.3

2. EC2 세팅

A. Docker 세팅

i. Docker 설치

```
# EC2 서버에서 실행

apt-get update
apt-get install sudo
apt-get install vim
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) stable"
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io
```

ii. Docker compose 설치

```
sudo apt install jq
VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r)
DESTINATION=/usr/bin/docker-compose
sudo curl -L https://github.com/docker/compose/releases/download/${VERSION}/docker-compose-$(uname -s)-$(uname -m) -o $DESTINATION
sudo chmod 755 $DESTINATION
```

iii. 도커 네트워크 생성

```
sudo docker network create cd_network
```

B. HTTPS 설정

i. Certbot 설치

```
sudo apt-get update
sudo snap install core; sudo snap refresh core
sudo snap install certbot --classic
```

ii. Nginx 설치 및 설정

```
sudo apt install nginx

# 설정
cd /etc/nginx/conf.d
sudo vim default.conf
```

iii. Certbot을 nginx에 적용 후 재시작

```
sudo certbot --nginx
```

```
sudo nginx

# 설정 변경시 재시작하는 명령어
sudo nginx -t
sudo service nginx restart
```

3. 프로퍼티 정의

A. Nginx Default 값 세팅

i. nginx.conf

```
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
    worker_connections 768;
    # multi_accept on;
}

http {

    ##
    # Basic Settings
    ##

    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
    # server_tokens off;

    # server_names_hash_bucket_size 64;
    # server_name_in_redirect off;

    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    ##
    # SSL Settings
    ##

    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 TLSv1.3; # Dropping SSLv3, ref: P00DLE
    ssl_prefer_server_ciphers on;

    ##
    # Logging Settings
    ##

    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    ##
    # Gzip Settings
    ##

    gzip on;

    # gzip_vary on;
    # gzip_proxied any;
    # gzip_comp_level 6;
    # gzip_buffers 16 8k;
    # gzip_http_version 1.1;
    # gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

    ##
    # Virtual Host Configs
    ##

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}

#mail {
```

```
# # See sample authentication script at:
# # http://wiki.nginx.org/ImapAuthenticateWithApachePhpScript
#
# # auth_http localhost/auth.php;
# # pop3_capabilities "TOP" "USER";
# # imap_capabilities "IMAP4rev1" "UIDPLUS";
#
# server {
#     listen    localhost:110;
#     protocol  pop3;
#     proxy     on;
# }
#
# server {
#     listen    localhost:143;
#     protocol  imap;
#     proxy     on;
# }
#}
```

ii. reverse proxy

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    autoindex_localtime on;
    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    # pass PHP scripts to FastCGI server
    #
    #location ~ /\.php$ {
    #    include snippets/fastcgi-php.conf;
    #
    #    # With php-fpm (or other unix sockets):
    #    fastcgi_pass unix:/var/run/php/php7.4-fpm.sock;
    #    # With php-cgi (or other tcp sockets):
    #    fastcgi_pass 127.0.0.1:9000;
    #}

    # deny access to .htaccess files, if Apache's document root
    # concurs with nginx's one
    #
    #location ~ /\.ht {
    #    deny all;
    #}
}
```

B. Git Ignore 파일 설정

```
# See https://help.github.com/articles/ignoring-files/ for more about ignoring files.

# dependencies
/node_modules
```

```
/.pnp
.pnp.js

# testing
/coverage

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*

# environment
.env
.DS_Store
models/
venv/
.vscode
```

4. Jenkins 세팅

A. Jenkins 구성

i. 젠킨스 이미지 다운로드 및 실행

```
sudo docker pull jenkins/jenkins:lts-jdk11

docker run -u 0 -d -p 9090:8080 -p 50000:50000 -v /var/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock --name j
```

ii. Gitlab, Docker 플러그인 설정

- Generic Webhook Trigger Plugin
- GitLab
- Gitlab API Plugin
- GitLab Authentication plugin
- Docker API Plugin
- Docker Commons Plugin
- Docker Compose Build Step Plugin
- Docker Pipeline
- Docker plugin
- docker-build-step

iii. jenkins에 docker 접근 설정

```
apt-get update
apt-get install \
    ca-certificates \
    curl \
    gnupg \
    sb-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 \
    $(sb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null
apt-get update
```

```
apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# docker-compose 사용 설정
sudo apt install jq
VERSION=$(curl --silent https://api.github.com/repos/docker/compose/releases/latest | jq .name -r)
DESTINATION=/usr/bin/docker-compose
sudo curl -L https://github.com/docker/compose/releases/download/${VERSION}/docker-compose-$(uname -s)-$(uname -m) -o $DESTINATION
sudo chmod 755 $DESTINATION
```

General

Enabled 

설명

[Plain text] [미리보기](#)

☒ Do not allow concurrent builds

☐ Abort previous builds [?](#)

☐ Do not allow the pipeline to resume if the controller restarts

☐ GitHub project

GitLab Connection

jenkins-token

☒ Use alternative credential

Credential :

GitLab API token (jenkins-token)

+ Add

Test Connection

- Jenkins를 Docker container로 띄운 후 프로젝트를 만들어 구성에 정보를 작성합니다.

Build Triggers

- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://jenkins.churros.site/project/A503> ?

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

5. 데이터베이스 세팅

A. Mariadb 설정

i. Dockerfile 작성

```
FROM mariadb:10.5.5

ENV MYSQL_DATABASE=churros
ENV MYSQL_ROOT_PASSWORD=A503

COPY ./config /etc/mysql/conf.d

EXPOSE 3306

CMD ["mysqld"]
```

ii. mariadb.cnf 작성

```
[client]
default-character-set=utf8mb4

[mysql]
default-character-set=utf8mb4

[mysqld]
character-set-server=utf8mb4
collation-server=utf8mb4_unicode_ci
skip-character-set-client-handshake

[mysqldump]
default-character-set=utf8mb4
```

iii. Dockerfile 빌드 및 실행


```
sudo docker build -t churros-mariadb .
sudo docker run --name churros-mariadb -d --network cd_network -v my-db-volume:/var/lib/mysql -e MARIADB_DATABASE=<사용할 db이름> -e MARIADB_ROOT_PASSWORD=<사용할 비밀번호>
```

B. Monbodb 설정

i. Dockerfile작성

```
FROM mongo:4.2

COPY ./mongod.conf /etc/mongod.conf

RUN /usr/bin/mongod --fork --logpath /var/log/mongodb.log --config /etc/mongod.conf && \
    sleep 10 && \
    mongo --eval 'rs.initiate({ _id: "churros", version: 1, members: [{ _id: 0, host: "localhost:27017" }]} )' && \
    sleep 10 && \
    mongo admin --eval 'db.getSiblingDB("admin").createUser({user: "admin", pwd: "admin123", roles: [{role: "root", db: "admin"}]})' && \
    sleep 10 && \
    mongo admin -u admin -p admin123 --eval 'db.getSiblingDB("newsdb").createUser({user: "churros", pwd: "A503", roles: [{role: "readwrite", db: "newsdb"}]})' && \
    sleep 10 && \
    mongo admin -u admin -p admin123 --eval 'db.getSiblingDB("newsdb").grantRolesToUser("churros", [{ role: "read", db: "local", collection: "churros" }])' && \
    /usr/bin/mongod --shutdown
EXPOSE 27017

CMD ["mongod", "--bind_ip_all", "--config", "/etc/mongod.conf"]
```

ii. mongod.conf 설정

```
security:
  authorization: enabled

replication:
  replSetName: churros
```

iii. Dockerfile 빌드 및 실행

```
sudo docker build -t churros-mongodb .
sudo docker run --name churros-mongodb -d --network cd_network -v mongodb:/data -e MARIADB_DATABASE=<사용할 db이름> -e MARIADB_ROOT_PASSWORD=<사용할 비밀번호>
```

C. Elasticsearch 설정

i. Dockerfile 작성

```
FROM docker.elastic.co/elasticsearch/elasticsearch:7.17.9

# 사용자 지정 설정 파일 추가
COPY ./config/elasticsearch.yml /usr/share/elasticsearch/config/
COPY ./config/jvm.options /usr/share/elasticsearch/config/

# Elasticsearch 플러그인 설치
RUN elasticsearch-plugin install analysis-nori
RUN /usr/share/elasticsearch/bin/elasticsearch-keystore create
RUN echo "A503!23" | /usr/share/elasticsearch/bin/elasticsearch-keystore add -x 'bootstrap.password'
```

ii. Dockerfile 빌드 및 실행

```
docker build -t churros-elasticsearch .

docker run -d --name churros-elasticsearch -p 9200:9200 -p 9300:9300 churros-elasticsearch
```

D. Monstache 설정

i. Dockerfile 작성

```
# Builder stage
FROM golang:1.17-alpine AS builder

RUN apk add --no-cache git gcc musl-dev

# Set the working directory
WORKDIR /app

# Download and install monstache (use specific version instead of latest)
RUN go get -u -d github.com/rwynn/monstache/v6

# Build monstache
RUN go install github.com/rwynn/monstache/v6@latest

# Final stage
FROM alpine:latest

# Install ca-certificates
RUN apk --no-cache add ca-certificates

# Copy binary from builder stage
COPY --from=builder /go/bin/monstache /usr/local/bin/

# COPY ./config /etc/monstache/config
CMD ["monstache", "-f", "/etc/monstache/config/config.toml"]
```

ii. conf/config.toml 작성

```
# 로깅 설정
verbose = true

# 연결 설정
mongo-url = "mongodb://churros:A503@churros-mongodb:27017"
elasticsearch-urls = ["http://churros-elasticsearch:9200"]
# elasticsearch-urls = ["https://localhost:9200"]

# 자주 필요한 설정
direct-read-namespaces = ["newsdb.newsCol"]
change-stream-namespaces = ["newsdb.newsCol"]

# 추가 설정
elasticsearch-user = "elastic"
elasticsearch-password = "A503!23"

# Routing 설정
[[mapping]]
namespace = "newsdb.newsCol"
index = "news"

# 필드 필터링 스크립트
[[script]]
namespace = "newsdb.newsCol"
script = """
module.exports = function(doc) {
  // 새로운 문서 객체 생성
  var newDoc = {
    _idx: doc._idx,
    link: doc.link,
    img_src: doc.img_src,
    title: doc.title,
    description: doc.description
  };

  // 새로운 문서 객체 반환
  return newDoc;
}
"""
```

iii. Dockerfile 빌드 및 실행

```
docker build -t churros-monstache .

docker run -d --name churros-monstache churros-monstache
```

6. DB 계정

1. `sudo docker exec -it churros-mariadb bash` 로 mariadb 컨테이너에 접속
2. `mysql -u root -p` 로 mariadb cli에 접속
3. `A503` 비밀번호 입력

A. mariaDB

- user : `root`
- passwd : `A503`
- database : `churros`

B. MongoDB

i. admin 계정

- user : `admin`
- passwd : `admin123`
- database : `admin`

ii. user 계정

- user : `churros`
- passwd : `A503`
- database : `newsdb`

C. Elasticsearch

- user :
- passwd :

7. 빌드 방법

A. 백엔드 빌드 방법

i. 데이터베이스 세팅

`RDBMS`, `elasticsearch`, `mongo`, `monstache` 를 각각 컨테이너로 띄운다.

ii. WAS Dockerfile 작성

```
# WAS Dockerfile
# openjdk:11-jdk-slim 을 이미지로 사용한다.
FROM openjdk:11-jdk-slim AS builder

# 작업 디렉토리를 /app으로 설정한다.
WORKDIR /app

# 모든 소스코드 (현재 폴더에 있는 소스코드) 를 작업 디렉토리에 복사한다.
COPY . .

# gradlew을 사용하여 build한다.
RUN chmod +x gradlew
RUN ./gradlew -x compileTestJava build

# 빌드가 완료된 jar파일을 작업 디렉토리의 root로 가져온다.
FROM openjdk:11-jdk-slim

WORKDIR /app

COPY --from=builder /app/build/libs/*.jar app.jar
```

```
# 9999 포트를 사용
EXPOSE 9999

# java -jar app.jar을 이용해서 빌드한 파일을 실행한다.
CMD ["java", "-jar", "-Dspring.profiles.active=devenv", "-Duser.timezone=Asia/Seoul", "app.jar"]
```

iii. CRS Dockerfile 작성

```
FROM python:3.9

ENV MARIA_USER=root
ENV MARIA_PASSWD=A503
ENV MARIA_HOST=churros-mariadb
ENV MARIA_PORT=3306
ENV MARIA_DB_NAME=churros

ENV MONGO_HOST=churros-mongodb
ENV MONGO_PORT=27017
ENV MONGO_USER=churros
ENV MONGO_PASSWD=A503
ENV MONGO_DB_NAME=newsdb
ENV MONGO_ADMIN_DB=admin

WORKDIR /fast-api

COPY requirements.txt requirements.txt

RUN pip install -r requirements.txt

COPY . .

CMD ["uvicorn", "app.api.main:app", "--host", "0.0.0.0", "--port", "8000"]
```

iv. 최상위 폴더에서 docker-compose up 실행

```
version: '3.8'
services:
  churros-backend:
    container_name: churros-backend
    build:
      dockerfile: Dockerfile
      context: ./backend
    environment:
      - MARIADB_PASSWORD=A503
      - MONGODB_URI=mongodb://churros:A503@churros-mongodb:27017/newsdb
    ports:
      - "9999:9999"
    volumes:
      - /etc/localtime:/etc/localtime:ro
  churros-crs:
    container_name: churros-crs
    build:
      context: ./recommend/pyrecommend-server
      dockerfile: Dockerfile
    ports:
      - "5555:8000"
    volumes:
      - /etc/localtime:/etc/localtime:ro
networks:
  default:
    name: cd_network
    external: true
```

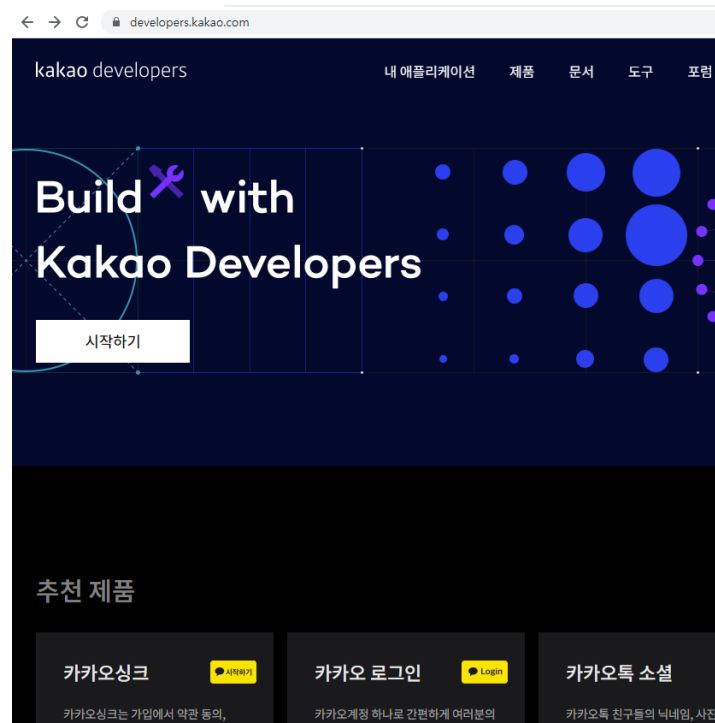
B. 프론트엔드 빌드 방법

```
npm i
npm start build
```

8. 외부 서비스

A. 카카오 로그인

- 카카오 개발자 페이지 접속 → 시작하기 버튼 누르기



어플리케이션 추가하기 버튼 누르기



애플리케이션 추가하기

앱 아이콘



파일 선택

JPG, GIF, PNG

권장 사이즈 128px, 최대 250KB

앱 이름

churros123

사업자명

churros123

- 입력된 정보는 사용자가 카카오 로그인을 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

그럼 다음과 같이 생성됨



churros123

ID 889220

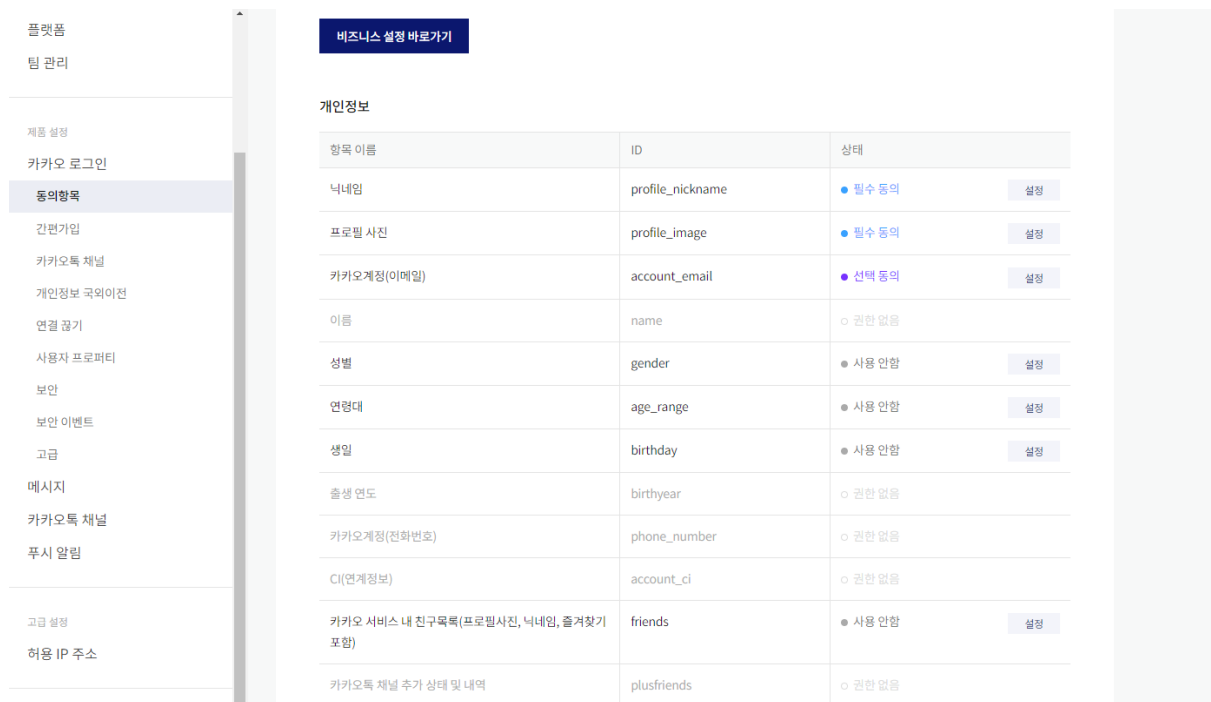
OWNER

앱 키

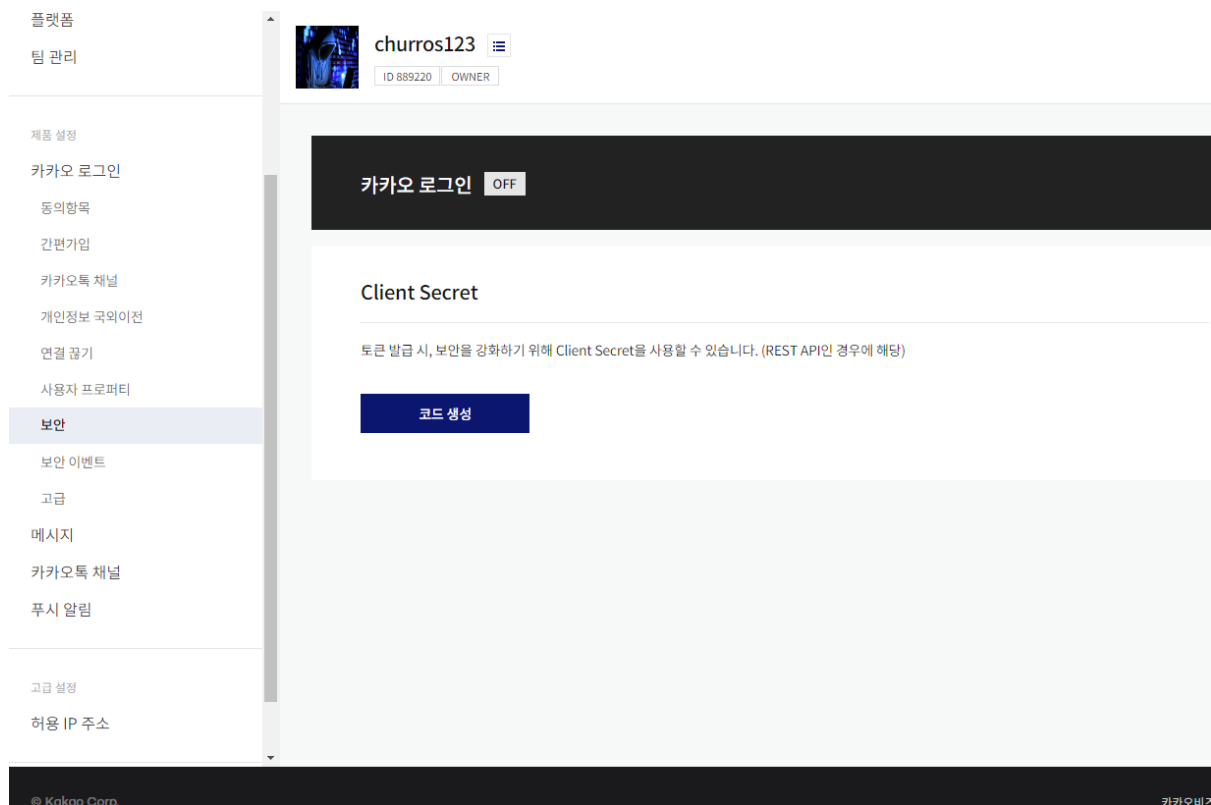
네이티브 앱 키	6c2fa438e4e7f609cd19696cfd1c8662
REST API 키	03f969273b2b1969093ba2d10639031e
JavaScript 키	8d7fa74bc92d400cfe64bbafe625dc4a
Admin 키	d3cbd929ff77715d493af90029648b2f

카카오 로그인- 동의 항목 페이지로 들어감

닉네임, 프로필 사진 - 필수동의 , 카카오계정- 선택동의(필수동의의 검수 받지 못하면 불가)

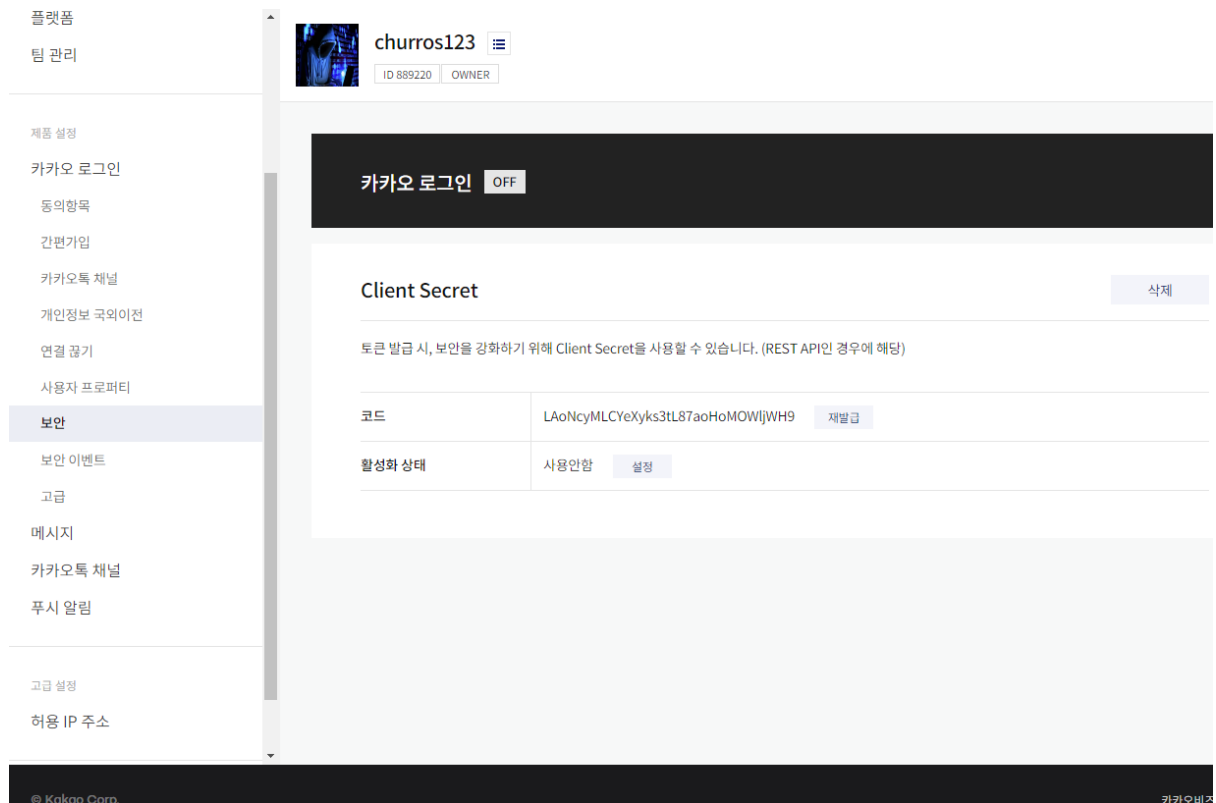


보안 - 코드생성 을 누른다.



다음과 같이 코드가 나온다.

코드가 바로 client-secret 이다.



application.yml 파일에 다음과 같이 넣어준다.

client-id는 앱키의 Rest-api 키 이고,

client-secret 은 보안 Client Secret의 코드이다.

```

kakao:
  client-id: 03f969273b2b1969093ba2d10639031e
  client-secret: LAoNcyMLCYeXyks3tL87aoHoMOWljWH9
  
```