

B206 : YOGIDICE

삼성 청년 SW아카데미 대전캠퍼스 7기

특화 프로젝트 (2022. 08. 22 ~ 2022. 10. 7)

포팅 매뉴얼

김현주(팀장), 김동신, 박정현, 전병찬, 최원재

목차

프로젝트 개요	3
프로젝트 기술 스택	3
CI/CD	5
빌드 상세내용	11
DB	14
외부 서비스	16

프로젝트 개요

YOGIDICE는 총 15000여 개의 게임과 2500만여 건의 유저 평가 데이터를 활용하여 사용자의 취향에 꼭 알맞는 게임을 추천해주는 서비스입니다. YOGIDICE는 사용자의 평가 내용을 기반으로 KNN알고리즘을 활용하여 비슷한 취향을 가진 다른 사용자들이 좋아하는 게임을 추천하고, Jaccard Similarity 알고리즘을 활용하여 플레이한 게임과 비슷한 유형의 다른 게임을 추천합니다.

프로젝트 기술 스택

가. 이슈 관리 : Jira

나. 형상 관리 : GitLab

다. 커뮤니케이션 : Mattermost, Notion

라. 개발환경

1) OS : Window 10

2) IDE

A. IntelliJ IDEA Ultimate

B. Visual Studio Code

C. UI/UX : Figma

3) Database : MySQL Workbench

4) Server : AWS EC2 (MobaXterm)

A. Ubuntu 20.04.4 LTS

B. Docker 20.10.17

C. Jenkins 2.346.2 LTS

D. Nginx 1.18.0

마. 상세 내용

1) Backend

- A. JAVA (Open JDK (Zulu 8.33.0.1))
- B. Spring Boot Gradle 7.5
- C. Lombok 1.18.20, Swagger3.0.0, JPA, QueryDSL

2) Frontend

- A. HTML, CSS, JavaScript
- B. Vue 3.0.0, Vuex 4.0.2
- C. Nodejs 16.16.0

CI/CD

CI/CD 작동 방식은 GitLab에서 Push Event가 발생하면 WebHook 기능을 통해 Jenkins에서 자동으로 빌드를 실행합니다. Jenkins에서 Nginx서버를 내포한 Vue, Spring Boot, Django 각 디렉토리 내부의 Dockerfile을 활용하여 Docker 이미지를 생성한 후 SSH 연결을 통해 AWS에 Docker Container를 생성하는 방식입니다.

1. MobaXterm을 이용해 EC2 서버 원격 접속.
2. Ubuntu환경에서 Docker 설치.
3. Docker-compose.yml을 만들고 Docker-compose up -d 명령어를 사용하여 Jenkins와 프로젝트에 사용할 MySQL 컨테이너를 생성.


```
version: '3'
services:
  jenkins:
    image: jenkins/jenkins:lts
    container_name: jenkins
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /jenkins:/var/jenkins_home
    ports:
      - "9090:8080"
    privileged: true
    user: root
  db:
    image: mysql:8.0.29
    container_name: mysql
    command:
      - --default-authentication-plugin=mysql_native_password
      - --character-set-server=utf8mb4
      - --collation-server=utf8mb4_unicode_ci
      - --range_optimizer_max_mem_size=16777216
    restart: always
    environment:
      MYSQL_DATABASE: yogidice
      MYSQL_USER: yogidice
      MYSQL_PASSWORD: 9x9sns99
      MYSQL_ROOT_PASSWORD: 9x9sns99
      TZ: Asia/Seoul
    volumes:
      # - {연결될 실제 물리 folder path}/{docker 안에 folder path}
      - ./mysql/data:/var/lib/mysql
      # OS 볼륨의 타임존을 따라가게 할 때 사용한다.
      - ./mysql/initdb.d:/docker-entrypoint-initdb.d
    ports:
      - 3306:3306
```


4. 새로운 Jenkins 프로젝트 생성 및 설정


Enter an item name


voaidicedeploy


» Required field

**Freestyle project**
이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다

**Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing

**Multi-configuration project**
다양한 환경에서의 테스트, 플레폼 특성 빌드, 기타 등등 처럼 다수의 서로다른 환경설정이 필요한 프로젝트에 적합함.

**Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate name as long as they are in different folders.

**Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

A. 구성에서 소스코드 관리 -> Git 설정에서 Git 레포지토리에 관한 설정 진행

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

https://lab.ssfy.com/s07-bigdata-recom-sub2/S07P22B206.git

Credentials ?

wj5295@naver.com/*****

+ Add

Name ?

Refspec ?

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

B. 빌드 유발 탭에서 아래와 같이 체크박스 체크. 그 후 고급 버튼을 클릭하고 Secret token을 찾아 Generate 버튼을 눌러 토큰을 생성. (GitLab과 WebHook 연결에 사용)

빌드 유발

- ☐ 빌드를 원격으로 유발 (예: 스크립트 사용) ?
- ☐ Build after other projects are built ?
- ☐ Build periodically ?
- ☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://j7b206.p.ssafy.io:9090/project/yogidicedeploy_ ?
 - Enabled GitLab triggers
 - ☒ Push Events
 - ☐ Push Events in case of branch delete
 - ☒ Opened Merge Request Events

Secret token ?

b77deda04094746641db8d550704f9f1

Generate

Clear

C. Steps에서 Execute shell을 추가하여 다음과 같이 입력 후 저장.

≡ Execute shell ?

Command

See [the list of available environment variables](#)

```
docker image prune -a --force
mkdir -p /var/jenkins_home/images_tar
cd /var/jenkins_home/workspace/yogidicedeploy/frontend/yogidice/
docker build -t vue .
docker save vue > /var/jenkins_home/images_tar/vue.tar

cd /var/jenkins_home/workspace/yogidicedeploy/analyze/
docker build -t django .
docker save django > /var/jenkins_home/images_tar/django.tar

cd /var/jenkins_home/workspace/yogidicedeploy/yogidice/
docker build -t spring .
docker save spring > /var/jenkins_home/images_tar/spring.tar

ls /var/jenkins_home/images_tar
```

5. 자동 빌드를 위한 GitLab WebHook 연결.

- A. 배포할 프로젝트의 GitLab 레포지토리에서 Settings-> Webhooks 페이지로 이동 후 URL에는 `http://배포서버공인IP:9090/project/{생성한jenkins프로젝트명}/`을 입력, Secret token에는 Jenkins 프로젝트 생성 시 발행한 Secret token 입력.

Q Search page

Webhook

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.

☐ Confidential issues events
 A confidential issue is created, updated, closed, or reopened.

☒ Merge request events
 A merge request is created, updated, or merged.

B. Webhook을 생성 후 test로 정상 작동하는지 확인.

6. 젠킨스 SSH 연결 설정 (Publish over SSH)

A. Dashboard -> Jenkins 관리 -> 시스템 설정의 Publish over SSH탭으로 이동

B. SSH Servers 탭에 추가 버튼 클릭

C. Name, Hostname, Username을 아래와 같이 입력 후 고급 버튼 클릭.

SSH Servers

SSH Server
 Name ?

Hostname ?

Username ?

Remote Directory ?

 고급...

- D. Use password authentication, or use different key 체크 후, 제공 받은 pem 파일의 내용을 복사 붙여 넣기.

☒ Use password authentication, or use a different key ?

Passphrase / Password ?

Change Password

Path to key ?

Key ?

7. 빌드 후 조치

- A. 프로젝트 구성 페이지의 빌드 후 조치 추가 클릭, Send build artifacts over SSH를 선택
- B. 빌드 후 실행할 명령문을 다음과 같이 작성

Send build artifacts over SSH ?

SSH Publishers

SSH Server Name ?

고급...

Transfers

Transfer Set Source files ?

Exec command ?

```
sudo docker load < /jenkins/images_tar/vue.tar
sudo docker load < /jenkins/images_tar/django.tar
sudo docker load < /jenkins/images_tar/spring.tar

if (sudo docker ps | grep "vue"); then sudo docker stop vue; fi
if (sudo docker ps | grep "django"); then sudo docker stop django; fi
if (sudo docker ps | grep "spring"); then sudo docker stop spring; fi

sudo docker run -it -d --rm -p 80:80 -p 443:443 -v /home/ubuntu/certbot/conf:/etc/letsencrypt/ -v /home/ubuntu/certbot/www:/var/www/certbot --name vue vue
echo "Run vue"
sudo docker run -it -d --rm -p 8000:8000 --name django django
echo "Run django"
sudo docker run -it -d --rm -p 8081:8081 --name spring spring
echo "Run Spring"
```

8. HTTPS 적용 및 Nginx 설정

- A. 다음 명령어를 통해 Certbot container를 생성하고 인증서를 발급

```
sudo docker run -it --rm --name certbot -p 80:80\  
-v "/home/ubuntu/certbot/conf:/etc/letsencrypt" \  
-v "/home/ubuntu/certbot/log:/var/log/letsencrypt" \  
-v "/home/ubuntu/certbot/www:/var/www/certbot" \  
certbot/certbot certonly
```

- B. standalone를 선택하고 이메일, 도메인 이름 등을 차례로 입력
- C. ubuntu환경에서 Vue 프로젝트 폴더로 이동 후 deploy_conf 디렉토리를 생성하고 그 안에서 nginx.conf 파일을 생성(설정 내용은 빌드 상세 내용 확인).

빌드 상세내용

Docker-compose와 Dockerfile 파일을 이용해 빌드합니다.

- Docker-compose.yml

```
version: '3'

services:
    # 컨테이너내의 서비스
    jenkins:
        # 서비스명: jenkins
        image: jenkins/jenkins:lts
        # 컨테이너 생성 시 jenkins/jenkins:lts 이미지를 기반 이미지로 사용
        container_name: jenkins
        # 컨테이너 이름
        volumes:
            # /var/run/docker.sock:/var/run/docker.sock # /var/run/docker.sock와 컨테이너 내부의 /var/run/docker.sock를 연결
            # /jenkins:/var/jenkins_home # /jenkins 폴더와 /var/jenkins_home 폴더를 연결
        ports:
            # 9090:8080
            # 포트 매핑, Aws의 9090 포트와 컨테이너의 8080 포트를 연결
        privileged: true
        # 컨테이너 시스템의 주요 자원에 연결할 수 있게 설정(기본적으로 false)
        user: root # 젠킨스에 접속할 유저 계정 (root로 할 경우 관리자)
    # 서비스 명: db
    db:
        image: mysql:8.0.29
        # 컨테이너 생성 시 mysql:8.0.29 이미지를 기반 이미지로 사용
        container_name: mysql
        command:
            # 아래의 4줄 : mysql 설정 부분. 인증 방식, encoding 방식 설정
            --default-authentication-plugin=mysql_native_password
            --character-set-server=utf8mb4
            --collation-server=utf8mb4_unicode_ci
            --range_optimizer_max_mem_size=16777216
        restart: always # 컨테이너 항상 재시작
        environment:
            MYSQL_DATABASE: yogidice
            MYSQL_DATABASE: yogidice
            # 사용하는 db
            MYSQL_USER: yogidice
            # root 패스워드
            MYSQL_PASSWORD: 9x9sns99
            # 접속하는 user
            MYSQL_ROOT_PASSWORD: 9x9sns99
            # user 패스워드
            TZ: Asia/Seoul
        volumes:
            - ./mysql/data:/var/lib/mysql
            # 데이터 마운트 용도, ./mysql/data와 컨테이너 내의 /var/lib/mysql 연결
        ports:
            # 3306:3306
            # 포트 매핑, Aws의 3306 포트와 컨테이너의 3306 포트 연결
```

- Spring Boot Dockerfile(yogidice/Dockerfile)

```
FROM openjdk:8-jdk-alpine as build-stage
# openjdk:8-jdk-alpine as build-stage 이미지를 build-stage라고 지정

COPY gradlew .
# gradlew을 이미지에 복사
COPY gradle gradle
# gradle을 이미지에 복사
COPY build.gradle .
# build.gradle을 이미지에 복사
COPY settings.gradle .
# settings.gradle을 이미지에 복사
COPY src src
# spring 어플리케이션의 소스 코드를 이미지에 복사
RUN chmod +x ./gradlew
# gradlew 실행할 수 있는 권한 부여 (없으면 permission denied 발생)
RUN ./gradlew bootJar
# gradlew를 사용해서 실행 가능한 jar 파일 생성

# Deploy Stage
FROM openjdk:8-jdk-alpine
# 새로 생성할 이미지의 기반 이미지 지정(openjdk:8-jdk-alpine)

WORKDIR /var/jenkins_home/workspace/yogidicedeploy/yogidice
# working directory를 /var/jenkins_home/workspace/yogidicedeploy/yogidice로 지정
COPY --from=build-stage build/libs/*.jar app.jar
# build-stage 이미지에서 build/libs/*.jar 파일을 app.jar로 복사

EXPOSE 8080
ENTRYPOINT [ "java", "-jar", "/var/jenkins_home/workspace/yogidicedeploy/yogidice/app.jar" ] # 컨테이너 생성 후 최초 실행할 때 명령어 지정
```

- Django Dockerfile(analyze/Dockerfile)

```
FROM python:3.7.9 # 새로 생성할 이미지의 기반 이미지 지정(python:3.7.9)
WORKDIR /var/jenkins_home/workspace/yogidicedeploy/analyze # working directory를 /var/jenkins_home/workspace/yogidicedeploy/analyze로 지정
COPY requirements.txt ./ # requirements.txt를 이미지에 복사

RUN pip install --upgrade pip # pip 업데이트 실행
RUN pip install -r requirements.txt # requirements.txt의 모듈 install
COPY . . # 이미지로 모두 복사
CMD ["gunicorn", "analyze.wsgi", "--bind", "0.0.0.0:8000"] # 컨테이너 생성 후 실행한 명령어 지정
```

- Vue Dockerfile(frontend/yogidice/Dockerfile)

```
FROM node:16.16.0 as build-stage # node:16.16.0 이미지를 build-stage로 지정

WORKDIR /var/jenkins_home/workspace/yogidicedeploy/frontend/yogidice # working directory를 /var/jenkins_home/workspace/yogidicedeploy/frontend/yogidice 로 지정

COPY package*.json ./ # package*.json를 이미지에 복사

RUN npm install # npm install로 필요한 모듈 install

COPY . . # 모두 복사

RUN npm run build # npm run build로 build 파일 생성

FROM nginx:stable-alpine as production-stage # nginx:stable-alpine 이미지를 production-stage로 지정

COPY --from=build-stage /var/jenkins_home/workspace/yogidicedeploy/frontend/yogidice/dist /usr/share/nginx/html # build-stage의 {working directory}/dist에 있는 파일을 /usr/shae/nginx/html에 복사

COPY --from=build-stage /var/jenkins_home/workspace/yogidicedeploy/frontend/yogidice/deploy_conf/nginx.conf /etc/nginx/conf.d/default.conf # {working directory}/deploy_conf의 nginx.conf파일을 /etc/nginx/conf.d/default.conf로 복사

EXPOSE 80 # 80번 포트로 개방
CMD ["nginx", "-g","daemon off;"] # 컨테이너 실행되면 nginx 실행, global 디렉티브로 daemon off 옵션 적용
```

- Nginx 설정

```
upstream backend{
    ip_hash;
    server 172.18.0.1:8081;
}

upstream analyze{
    ip_hash;
    server 172.18.0.1:8000;
}

map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

server {
    listen 80;
    server_name yogidice.site;
    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name yogidice.site;
    access_log /var/log/nginx/access.log;
    error_log /var/log/nginx/error.log;

    ssl_certificate /etc/letsencrypt/live/yogidice.site/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yogidice.site/privkey.pem;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2 SSLv3;
    ssl_ciphers ALL;

    location /api/ {
        proxy_pass http://backend;
        proxy_redirect off;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $http_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-Nginx-Proxy true;
    }

    location /analyze/ {
        proxy_pass http://analyze;
        proxy_redirect off;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection $connection_upgrade;
        proxy_set_header Host $http_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-Nginx-Proxy true;
    }

    location / {
        root /usr/share/nginx/html;
        index index.html index.htm;
        try_files $uri $uri/ /index.html;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        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-Nginx-Proxy true;
    }

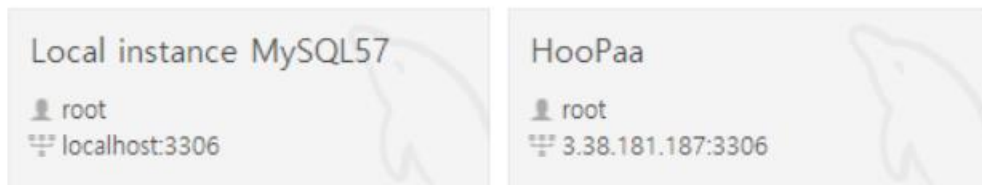
    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
        root /usr/share/nginx/html;
    }
}
```

80번 포트(http)로 들어온 요청은 443번 포트(https)로 리다이렉트 시킨다. url 에 /api가 붙어 있는 요청은 Spring Boot서버로 연결한다.

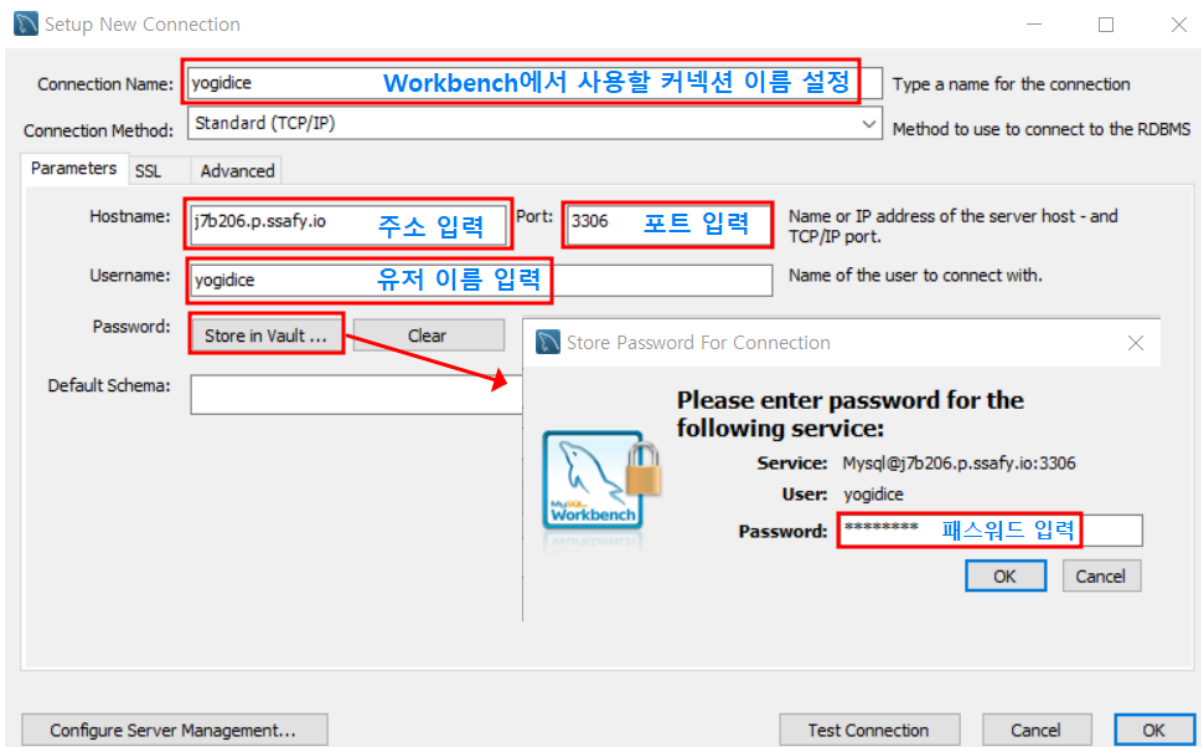
DB

1. MySQL Workbench 추가

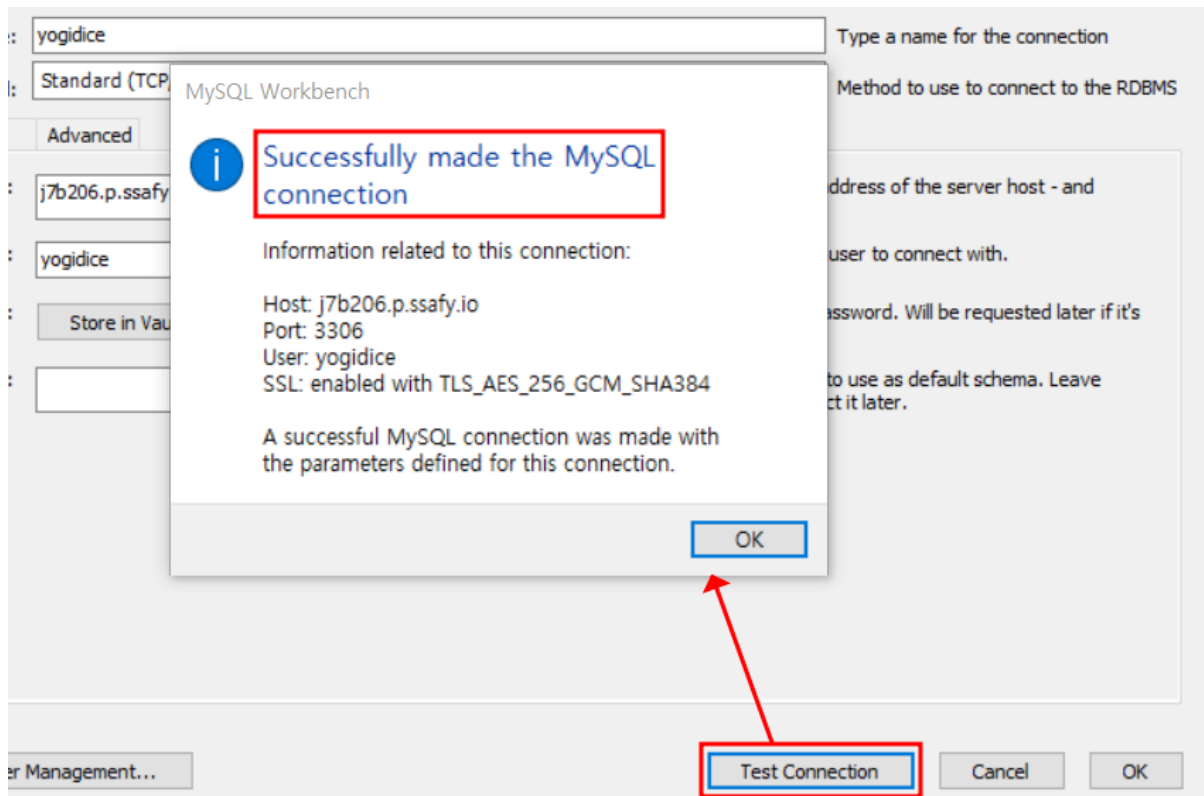
MySQL Connections



2. EC2 계정 정보 작성

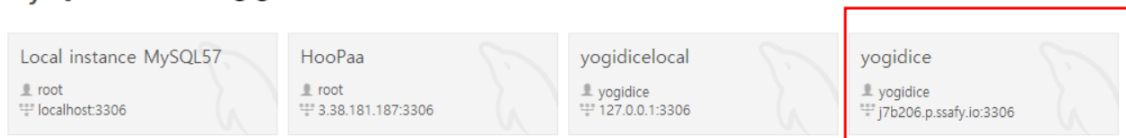


3. 테스트 커넥션 확인



4. 커넥션 추가 확인

MySQL Connections + -



외부 서비스

가비아에서 도메인 네임을 구입하여 가비아의 DNS에 등록하였습니다.

yogidice.site

• 레코드 개수 : 2개 • 최근 업데이트 : 2022-09-23 00:21:39 • 네임서버 : ns.gabia.co.kr

DNS 설정

레코드 수정

타입	호스트	값/위치
A	www	54.180.134.48
A	@	54.180.134.48

서비스의 편리한 이용을 위해 카카오 로그인/회원가입 기능을 적용하였습니다. 부가적인 서비스 제공을 위해 카카오 서버를 통해 사용자의 정보(닉네임, 카카오 계정(이메일))를 가져온 뒤 필요한 정보를 추가로 입력 받아 회원가입 하는 방식을 사용하였습니다.

1. 어플리케이션 추가

애플리케이션 추가하기

앱 아이콘



파일 선택
JPG, GIF, PNG
권장 사이즈 128px, 최대 250KB

앱 이름

요기다이스

사업자명

전병찬

• 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.

• 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

취소

저장

2. 도메인 등록

Web

[삭제](#)[수정](#)

사이트 도메인	https://i7b307.p.ssafy.io
---------	---------------------------

- 카카오 로그인 사용 시 Redirect URI를 등록해야 합니다. [등록하러 가기](#)

3. Redirect URI 설정

Redirect URI

[삭제](#)[수정](#)

Redirect URI	https://i7b307.p.ssafy.io/kakaologin
--------------	--------------------------------------

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

4. 로그인 활성화

카카오 로그인 ☒ ON

[동의 화면 미리보기](#)

활성화 설정

상태

☒ ON





카카오 로그인 API를 활용하면 사용자들이 번거로운 회원 가입 절차 대신, 카카오톡으로 서비스를 시작할 수 있습니다.

상태가 OFF일 때도 카카오 로그인 설정 항목을 변경하고 서버에 저장할 수 있습니다.

상태가 ON일 때만 실제 서비스에서 카카오 로그인 화면이 연결됩니다.

5. 카카오 인가 코드 수신

앱 키

플랫폼	앱 키	재발급
네이티브 앱 키		복사 재발급
REST API 키		복사 재발급
JavaScript 키		복사 재발급
Admin 키		복사 재발급

- 네이티브 앱 키: Android, iOS SDK에서 API를 호출할 때 사용합니다.
- JavaScript 키: JavaScript SDK에서 API를 호출할 때 사용합니다.
- REST API 키: REST API를 호출할 때 사용합니다.
- Admin 키: 모든 권한을 갖고 있는 키입니다. 노출이 되지 않도록 주의가 필요합니다.

6. Kakao Access Token 수신

Request

URL

```
POST /oauth/token HTTP/1.1
Host: kauth.kakao.com
Content-type: application/x-www-form-urlencoded;charset=utf-8
```

7. Access Token으로 사용자 정보 가져오기

Request: 액세스 토큰 사용

URL

```
GET/POST /v2/user/me HTTP/1.1
Host: kapi.kakao.com
Authorization: Bearer ${ACCESS_TOKEN}
Content-type: application/x-www-form-urlencoded;charset=utf-8
```