



# 손:걸음 포팅 매뉴얼

## 1. 개발환경

- [1.1. Frontend](#)
- [1.2. Backend](#)
- [1.3. Server](#)
- [1.4. Database](#)
- [1.5. UI/UX](#)
- [1.6. IDE](#)
- [1.7. 형상 / 이슈관리](#)
- [1.8. 기타 툴](#)

## 2. 환경변수

- [2.1. Frontend](#)
- [2.2 Backend](#)

## 3. EC2 세팅

- [3.1. Docker Engine 설치](#)
- [3.2. Nginx reverse proxy 설정 + SSL 인증서 발급](#)
- [3.3. EC2 포트 정리](#)
- [3.4. 방화벽\(UFW\) 설정](#)

## 4. 빌드 및 배포

- [4.1. OpenVidu](#)
- [4.2. Frontend \(Nginx / React\)](#)
- [4.3. Backend \(Spring Boot\)](#)
- [4.4. Backend \(Django\)](#)

## 5. DB (AWS RDS) 설정

- [5.1. MySQL Database 생성](#)
- [5.2. VPC 보안 그룹 인바운드 설정](#)
- [5.3. Spring Boot 연결](#)
- [5.4. MySQL Workbench 연결](#)

## 6. CI/CD 구축

- [6.1. Jenkins 도커 이미지 + 컨테이너 생성](#)
- [6.2. Jenkinsfile 생성](#)
- [6.3. Docker Compose 파일 생성](#)
- [6.4. build.sh 및 deploy.sh 파일 생성](#)
- [6.5. Jenkins 설정](#)
  - [6.5.1. Jenkins Plugin 추가설치](#)
  - [6.5.2. GitLab Credentials 설정](#)
  - [6.5.3. Jenkins Item 생성](#)
  - [6.5.4. GitLab Webhook 설정](#)
  - [6.5.5. 빌드 및 배포](#)

## 7. 외부서비스

- [7.1. 소셜 로그인 - Kakao](#)
  - [7.1.1. 애플리케이션 생성](#)
  - [7.1.2. 키 생성, 동의 항목 선택](#)
  - [7.1.3. application-oauth.yml 작성](#)
  - [7.1.4. 카카오톡부터 사용자 정보 얻어오기](#)
- [7.2. 통합 수어 정보](#)
- [7.3. 클라우드 관계형 데이터베이스](#)

## 1. 개발환경

### 1.1. Frontend

- Node.js 18.13.0 (LTS)
- React 18.2.0
  - Redux 4.2.1

- mui/material 5.11.6
- axios 1.2.6
- Sass 1.57.1
- Openvidu Browser 2.24.0
- jQuery 3.6.3

## 1.2. Backend

- Java
  - Java OpenJDK 1.8.0
  - Spring Boot 2.7.7
    - Spring Data JPA 2.7.6
    - Spring Security 5.7.6
    - JUnit 4.13.2
    - Lombok 1.18.24
    - Swagger 3.0.0
  - Gradle 7.6
- Python
  - Python 3.8.10
  - Django 4.1.5
  - OpenCV 4.5.5.64
  - MediaPipe 0.9.0.1

## 1.3. Server

- Ubuntu 20.04 LTS
- Nginx 1.18.0
- Docker 20.10.23
- Docker Compose 2.15.1
- OpenVidu 2.24.0

## 1.4. Database

- MySQL (AWS RDS) 8.0.30

## 1.5. UI/UX

- Figma 93.4.0

## 1.6. IDE

- Visual Studio Code 1.75
- IntelliJ IDEA 2022.3.1
- PyCharm 22.3.2

## 1.7. 형상 / 이슈관리

- Gitlab
- Jira

## 1.8. 기타 툴

- Postman 10.9.4
- Termius 7.56.1

## 2. 환경변수

### 2.1. Frontend

```
REACT_APP_API
REACT_APP_KAKAO_API
REACT_APP_KAKAO_CLIENT_ID
REACT_APP_KAKAO_CLIENT_SECRET
REACT_APP_KAKAO_REDIRECT_URI
```

### 2.2 Backend

```
SPRING_DATASOURCE_URL
SPRING_DATASOURCE_USERNAME
SPRING_DATASOURCE_PASSWORD
OPENVIDU_SECRET
JWT_SECRET
APP_AUTH_TOKENSECRET
APP_AUTH_TOKENEXPIRY
APP_AUTH_REFRESHTOKENEXPIRY
APP_OAUTH2_AUTHORIZEDREDIRECTURIS
SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_CLIENTID
SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_CLIENTSECRET
SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_CLIENTNAME
SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_AUTHORIZATIONGRANTTYPE

SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_REDIRECTURI
SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_CLIENTAUTHENTICATIONMI

SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_SCOPES_0
SPRING_SECURITY_OAUTH2_CLIENT_REGISTRATION_KAKAO_SCOPES_1
```

## 3. EC2 세팅

### 3.1. Docker Engine 설치

```
# 1. 구 버전 삭제
sudo apt-get remove docker docker-engine docker.io containerd runc

# 2. Repository 설정
sudo apt-get update
sudo apt-get install \
    ca-certificates \
    curl \
```

```

gnupg \
lsb-release
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo 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/ubuntu \
$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 3. Docker Engine 설치
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

# 4. 정상작동 확인
sudo docker run hello-world

```

## 3.2. Nginx reverse proxy 설정 + SSL 인증서 발급

```

# 1. Nginx 설치
sudo apt-get install nginx
nginx -v

# 2. Let's Encrypt 설치 및 SSL 발급
sudo apt-get install letsencrypt
sudo systemctl stop nginx
sudo letsencrypt certonly --standalone -d 도메인명

# 3. Nginx 설정파일 생성
cd /etc/nginx/sites-available
vi configure
#####

### DDos 방어
# 시간 당 request 비율 제한 (클라이언트 IP에 대한 요청 1초에 최대 5개)
limit_req_zone $binary_remote_addr zone=ddos_req:10m rate=5r/s;

server {

    location /jenkins/ {
        proxy_pass http://localhost:8081/jenkins/;
        proxy_redirect off;

        limit_req zone=ddos_req burst=10;

        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $host:$server_port;
        proxy_set_header X-Forwarded-Proto http;
        proxy_set_header X-Forwarded-Port "443";
        proxy_set_header X-Forwarded-Host $http_host;
    }

    location / {
        proxy_pass http://localhost:3000;
        limit_req zone=ddos_req burst=10;
    }

    location /api {
        proxy_pass http://localhost:8080/api;
        limit_req zone=ddos_req burst=10;
    }

    location /ws {
        proxy_pass http://localhost:8000/ws;
        limit_req zone=ddos_req burst=10;

        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
        proxy_set_header Host $host;
    }

    listen 443 ssl;
    ssl_certificate /etc/letsencrypt/live/i8b106.p.ssafy.io/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/i8b106.p.ssafy.io/privkey.pem;
}

server {

    if ($host = i8b106.p.ssafy.io) {
        return 301 https://$host$request_uri;
    }
}

```

```

        listen 80;
        server_name i8b106.p.ssafy.io;

        return 404;
    }

#####

sudo ln -s /etc/nginx/sites-available/configure /etc/nginx/sites-enabled/configure

sudo nginx -t # ok 시 성공

sudo systemctl restart nginx

```

### 3.3. EC2 포트 정리

Port 번호	내용
22	SSH
80	HTTP (HTTPS로 redirect)
443	HTTPS
3000	Nginx, React (Docker)
3478	TURN/STUN
8000	Django (Docker)
8080	Spring Boot (Docker)
8081	Jenkins
8443	OpenVidu

### 3.4. 방화벽(UFW) 설정

```

# 1. 해당 포트 개방
# 22 TCP
# 80 TCP
# 443 TCP
# 3000 TCP
# 3478 TCP+UDP
# 8000 TCP
# 8080 TCP
# 8081 TCP
# 40000-57000 TCP+UDP
# 57001-65535 TCP+UDP
# 8443 TCP (OpenVidu HTTPS 포트)
# 예시
sudo ufw allow 22/tcp

# 2. Firewall 활성화 / 상태 확인
sudo ufw enable
sudo ufw status verbose

# 3. Nginx reverse proxy 설정 후 Frontend, Backend, Jenkins 서버 포트 닫기
sudo ufw deny 3000/tcp # React Nginx
sudo ufw deny 8000/tcp # Django
sudo ufw deny 8080/tcp # Spring Boot
sudo ufw deny 8081/tcp # Jenkins

```

## 4. 빌드 및 배포

### 4.1. OpenVidu

```

# 1. OpenVidu 서버 생성
sudo su
cd /opt
curl https://s3-eu-west-1.amazonaws.com/aws.openvidu.io/
install_openvidu_latest.sh | bash

# 2. OpenVidu 서버 설정

```

```

cd openvidu
vi .env
./openvidu start

# 2-1. .env 파일 설정
# DOMAIN_OR_PUBLIC_IP=도메인명
# OPENVIDU_SECRET=내 비밀번호
# CERTIFICATE_TYPE=letsencrypt
# (이렇게 해야 SSL 인증서 생성가능; 하지만 도메인이 있어야함, IP주소 안됨)
# LETSENCRYPT_EMAIL=내 이메일
# HTTPS_PORT=8443

# 3. 테스트
# https:내도메인:8443/dashboard/ 접속
# user: OPENVIDUAPP / pass: 설정한 OPENVIDU_SECRET 값

# 4. HTTP_PORT 변경 (최초 SSL 인증서 발급 후 가능)
# vi .env
# HTTP_PORT=8442 (기본 http(80), https(443) 포트는 향후
# nginx reverse proxy server를 위해 비워둬야 함)

```

## 4.2. Frontend (Nginx / React)

```

# 1. Home Directory로 이동
cd /home/ubuntu

# 2. Git Clone
git clone 내깃랩

# 3. 위치 이동
cd 프로젝트폴더/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
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;"]

# 6. npm 모듈 설치
npm install --legacy-peer-deps

# 7. 빌드 파일 생성
npm run build

# 8. 도커 이미지 생성
docker build -t sgr-frontend:0.1 .
docker rm -f sgr-front # 기존 돌아가고 있는 컨테이너 있다면 제거
docker image prune # Dangling image 삭제

# 9. 도커 컨테이너 생성
docker run --name sgr-front -d -p 3000:80 sgr-frontend:0.1

```

## 4.3. Backend (Spring Boot)

```

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

# 2. Dockerfile 생성
FROM openjdk:8-jdk-alpine
VOLUME /tmp
ARG JAR_FILE=build/libs/bbb-0.0.1-SNAPSHOT.jar

```

```

COPY ${JAR_FILE} app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "/app.jar"]
ARG DEBIAN_FRONTEND=noninteractive

# 3. gradle 빌드파일 생성
gradle clean build -x test (gradle 설치 필요 / SDKMAN 패키지 매니저 사용 권장)
# 또는
sudo chmod +x ./gradlew
./gradlew clean build -x test

# 4. 도커 이미지 생성
docker build -t sgr-backend:0.1 .
docker rm -f sgr-back
docker image prune

# 5. 도커 컨테이너 생성
docker run --name sgr-back -d -p 8080:8080 --env-file
/home/ubuntu/env-files/env-back.env sgr-backend:0.1

```

## 4.4. Backend (Django)

```

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

# 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
RUN mkdir /project
ADD . /project
WORKDIR /project
RUN pip install --upgrade pip
RUN pip install -r requirements.txt
RUN pip install opencv-python-headless
RUN python manage.py migrate
EXPOSE 8000
CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"]

# 3. 도커 이미지 생성
docker rmi sgr-django:0.1
docker build -t sgr-django:0.1 .
docker rm -f sgr-dj

# 4. 도커 컨테이너 생성
docker run --name sgr-dj -d -p 8000:8000 sgr-django:0.1

```

## 5. DB (AWS RDS) 설정

AWS에서 제공하는 클라우드 기반 관계형 데이터베이스입니다. 서비스의 데이터 보안성 및 안정성 확립과 메인 EC2와의 분리를 위해 외부 데이터베이스를 사용했습니다.

### 5.1. MySQL Database 생성

1. AWS 홈페이지 접속 및 로그인
2. 서비스 → RDS 선택
3. “데이터베이스 생성” 클릭
4. “MySQL” 선택

## 데이터베이스 생성

### 데이터베이스 생성 방식 선택 정보

☒ 표준 생성

가용성, 보안, 백업 및 유지 관리에 대한 옵션을 포함하여 모든 구성 옵션을 설정합니다.

☐ 손쉬운 생성

권장 모범 사례 구성을 사용합니다. 일부 구성 옵션은 데이터베이스를 생성한 후 변경할 수 있습니다.

### 엔진 옵션

#### 엔진 유형 정보

☐ Amazon Aurora



☒ MySQL



☐ MariaDB



☐ PostgreSQL



☐ Oracle



☐ Microsoft SQL Server



### 5. “프리 티어” 선택

#### 템플릿

해당 사용 사례를 충족하는 샘플 템플릿을 선택하세요.

☐ 프로덕션

고가용성 및 빠르고 일관된 성능을 위해 기본값을 사용하세요.

☐ 개발/테스트

이 인스턴스는 프로덕션 환경 외부에서 개발 용도로 마련되었습니다.

☒ 프리 티어

RDS 프리 티어를 사용하여 새로운 애플리케이션을 개발하거나, 기존 애플리케이션을 테스트하거나 Amazon RDS에서 실무 경험을 받을 수 있습니다. [정보](#)

### 6. DB 인스턴스 명, 사용자 및 비밀번호 설정



## 설정

### DB 인스턴스 식별자 정보

DB 인스턴스 이름을 입력하세요. 이름은 현재 AWS 리전에서 AWS 계정이 소유하는 모든 DB 인스턴스에 대해 고유해야 합니다.

DB 인스턴스 식별자는 대소문자를 구분하지 않지만 'mydbinstance'와 같이 모두 소문자로 저장됩니다. 제약: 1~60자의 영숫자 또는 하이픈으로 구성되어야 합니다. 첫 번째 문자는 글자여야 합니다. 하이픈 2개가 연속될 수 없습니다. 하이픈으로 끝날 수 없습니다.

### ▼ 자격 증명 설정

#### 마스터 사용자 이름 정보

DB 인스턴스의 마스터 사용자에 로그인 ID를 입력하세요.

1~16자의 영숫자. 첫 번째 문자는 글자여야 합니다.

#### ☐ AWS Secrets Manager에서 마스터 보안 인증 관리 - 신규

Secrets Manager에서 마스터 사용자 보안 인증을 관리합니다. RDS는 사용자 대신 암호를 생성하고 수명 주기 동안 이를 관리할 수 있습니다.

#### ☐ 암호 자동 생성

Amazon RDS에서 사용자를 대신하여 암호를 생성하거나 사용자가 직접 암호를 지정할 수 있습니다.

#### 마스터 암호 정보

제약 조건: 8자 이상의 인쇄 가능한 ASCII 문자. 다음은 포함할 수 없습니다. /(슬래시), '(작은따옴표), "(큰따옴표) 및 @(앳 기호).

#### 마스터 암호 확인 정보

## 7. 퍼블릭 액세스 “예”로 설정

### 퍼블릭 액세스 정보



예

RDS는 데이터베이스에 퍼블릭 IP 주소를 할당합니다. VPC 외부의 Amazon EC2 인스턴스 및 다른 리소스가 데이터베이스에 연결할 수 있습니다. VPC 내부의 리소스도 데이터베이스에 연결할 수 있습니다. 데이터베이스에 연결할 수 있는 리소스를 지정하는 VPC 보안 그룹을 하나 이상 선택합니다.



아니요

RDS는 퍼블릭 IP 주소를 데이터베이스에 할당하지 않습니다. VPC 내부의 Amazon EC2 인스턴스 및 다른 리소스만 데이터베이스에 연결할 수 있습니다. 데이터베이스에 연결할 수 있는 리소스를 지정하는 VPC 보안 그룹을 하나 이상 선택합니다.

## 8. 포트번호 설정

### 데이터베이스 포트 정보

데이터베이스가 애플리케이션 연결에 사용할 TCP/IP 포트입니다.

## 9. “데이터베이스 생성” 클릭

## 5.2. VPC 보안 그룹 인바운드 설정

1. RDS 홈 화면 → “데이터베이스” → 생성된 DB 클릭
2. “연결 & 보안” → “보안” → “VPC 보안 그룹” 클릭

## 연결 & 보안

### 엔드포인트 및 포트

엔드포인트

[Redacted]

### 네트워킹

가용 영역

ap-northeast-2b

### 보안

VPC 보안 그룹

[Redacted]

🟢 활성화

3. “인바운드 규칙” → “인바운드 규칙 편집” 클릭

세부 정보 | **인바운드 규칙** | 아웃바운드 규칙 | 태그

🔔 이제 Reachability Analyzer를 사용하여 네트워크 연결을 확인할 수 있습니다. Reachability Analyzer 실행 ×

인바운드 규칙 (3) 🔄 태그 관리 **인바운드 규칙 편집**

4. “규칙 추가” → “소스” → 접근가능 IP 설정

인바운드 규칙 정보

보안 그룹 규칙 ID	유형 정보	프로토콜 정보	포트 범위 정보	소스 정보	설명 - 선택 사항 정보
[Redacted]	MySQL/Aurora	TCP	3306	사용자 ...	[Redacted]
[Redacted]	MySQL/Aurora	TCP	3306	사용자 ...	[Redacted]
[Redacted]	MySQL/Aurora	TCP	3306	사용자 ...	[Redacted]

규칙 추가

## 5.3. Spring Boot 연결

1. “엔드포인트” 및 “포트” 속지

연결 & 보안 | 모니터링 | 로그 및 이벤트 | 구성 | 유지 관리 및 백업 | 태그

### 연결 & 보안

#### 엔드포인트 및 포트

엔드포인트

[Redacted]

포트

3306

#### 네트워킹

가용 영역

ap-northeast-2b

VPC

[Redacted]

#### 보안

VPC 보안 그룹

[Redacted]

🟢 활성화

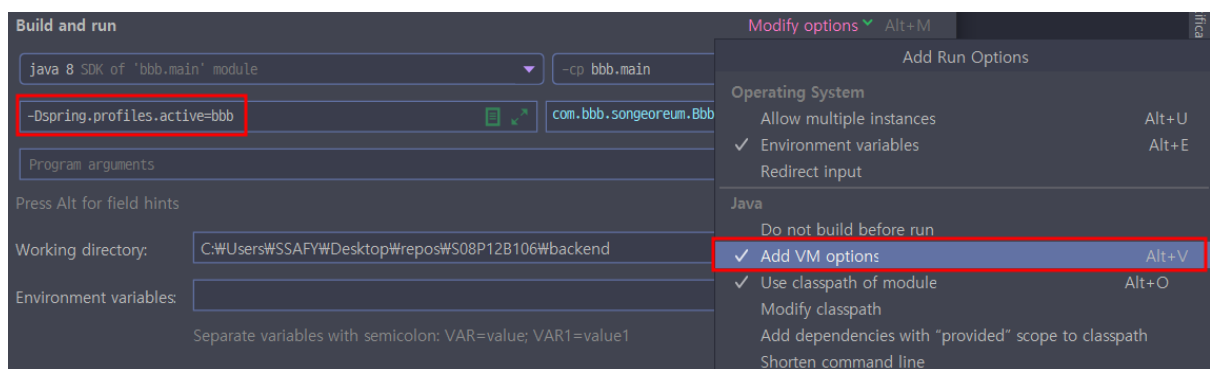
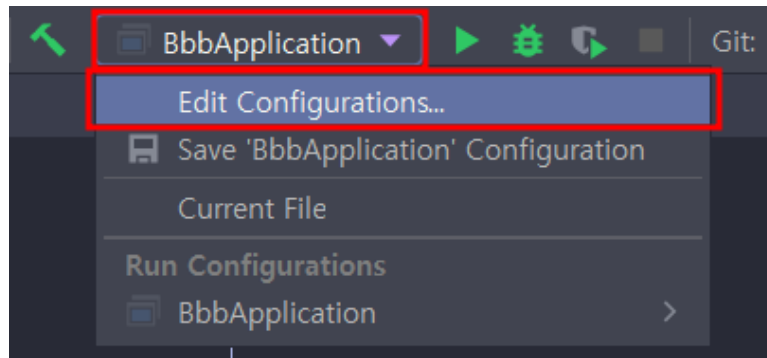
퍼블릭 액세스 가능

예

2. 로컬환경 : Spring Boot의 application-[임시명].yml에 DB 정보 입력

```
application-bbb.yml
1 # MySQL
2 spring:
3   datasource:
4     driver-class-name: com.mysql.cj.jdbc.Driver
5     url: [Redacted]
6     username: [Redacted]
7     password: [Redacted]
```

- ✓ 로컬환경에서의 사용을 위해 IntelliJ [프로젝트명]Application → Edit Configurations → Add VM Options 선택 →  
 Dspring.profiles.active=[위에 설정한 임시명] 설정 → .gitignore에 해당 yml 파일 추가

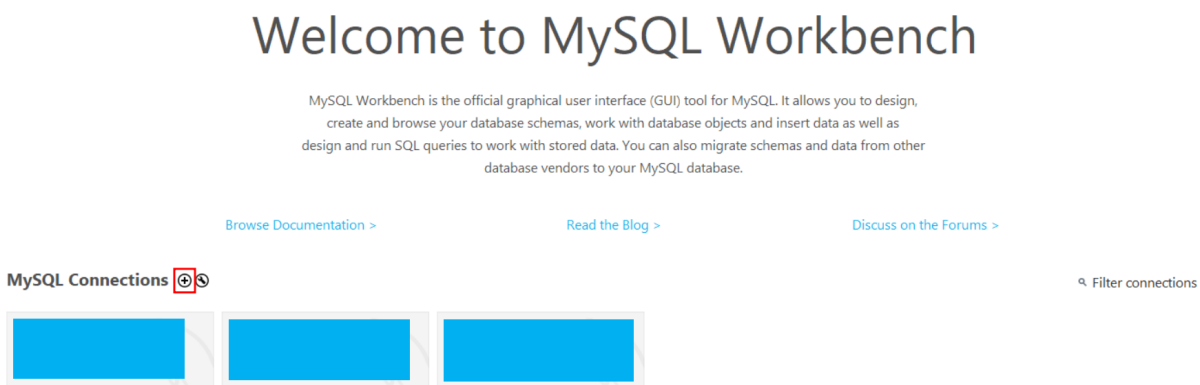


3. 배포환경 : EC2 내 .env 파일 생성하여 정보 입력 (.env 파일은 .gitignore에 추가)



## 5.4. MySQL Workbench 연결

1. MySQL 홈 화면 → "MySQL Connections +" 클릭



2. "Connection Name" : 임의 입력 / Hostname : 생성한 DB의 엔드포인트 입력 / Username : 생성한 DB의 user 입력 / Password :  
 "Store in Vault..." 클릭 후 생성한 DB의 password 입력

### 3. Connection 접속

## Welcome to MySQL Workbench


MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

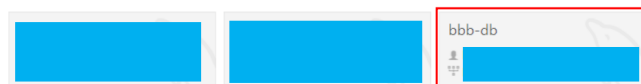
[Browse Documentation >](#)

[Read the Blog >](#)

[Discuss on the Forums >](#)

#### MySQL Connections

 Filter connections



## 6. CI/CD 구축

무중단 배포를 위해 Jenkins Pipeline을 구현하여 CI/CD 환경을 구축하였습니다. 또한 GitLab WebHooks 설정을 통해 master 브랜치로 push 시 자동으로 빌드와 배포를 진행하도록 설정하였습니다.

### 6.1. Jenkins 도커 이미지 + 컨테이너 생성

Jenkins Docker Container 생성을 위한 **Dockerfile** 생성 (Jenkins 컨테이너 위에 Docker 설치)

```
# Dockerfile

FROM jenkins/jenkins:ls

ENV DEBIAN_FRONTEND noninteractive
ENV DEBCONF_NOWARNINGS="yes"

USER root
RUN apt-get -y update && apt-get install -y --no-install-recommends \
    vim \
    apt-utils
```

```

RUN apt-get install ca-certificates curl gnupg lsb-release -y
RUN mkdir -p /etc/apt/keyrings
RUN curl -fsSL https://download.docker.com/linux/debian/gpg
| gpg --dearmor -o /etc/apt/keyrings/docker.gpg
RUN echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/debian $(lsb_release -cs) stable"
| tee /etc/apt/sources.list.d/docker.list > /dev/null
RUN apt-get -y update
RUN apt-get install docker-ce docker-ce-cli containerd.io
docker-compose docker-compose-plugin -y
RUN if [ -e /var/run/docker.sock ];
then chown jenkins:jenkins /var/run/docker.sock; fi
RUN usermod -aG docker jenkins
USER jenkins

```

## 6.2. Jenkinsfile 생성

Jenkins 파이프라인 구축을 위한 **Jenkinsfile** 생성

```

# Jenkinsfile

pipeline {
    agent any

    tools {
        gradle 'gradle_7.6'
        nodejs 'nodejs_18.13.0'
    }

    stages {
        stage('Build') {
            steps {
                sh "chmod +x -R ${env.WORKSPACE}"
                sh './pipeline/jenkins/build/build.sh'
            }
        }

        stage('Deploy') {
            steps {
                sh "chmod +x -R ${env.WORKSPACE}"
                sh './pipeline/jenkins/deploy/deploy.sh'
            }
        }
    }
}

```

## 6.3. Docker Compose 파일 생성

도커 이미지 빌드 및 컨테이너 실행 자동화 설정의 편리성을 위해 **docker-compose** 파일 생성

```

# docker-compose.yml

version: "3"
services:
    sgr-back:
        container_name: sgr-back
        image: sgr-backend:0.1
        build:
            context: ../backend
        ports:
            - "8080:8080"
        env_file:
            - ./env-files/env-back.env

    sgr-front:
        container_name: sgr-front
        image: sgr-frontend:0.1
        build:
            context: ../frontend
        ports:
            - "3000:80"

    sgr-dj:
        container_name: sgr-dj
        image: sgr-django:0.1
        build:
            context: ../backend-websocket

```

```
ports:
  - "8000:8000"
```

4. 빌드 및 배포 시 수행 명령을 `build.sh` 및 `deploy.sh` 셸 스크립트 파일 생성

## 6.4. build.sh 및 deploy.sh 파일 생성

```
# build.sh

#!/bin/bash

echo "*****"
echo "*****Building FRONTEND*****"
echo "*****"

# frontend 빌드
cd /var/jenkins_home/workspace/bbb-pipeline/frontend
npm install
npm run build

echo "*****"
echo "*****Building BACKEND*****"
echo "*****"

# backend 빌드
cd /var/jenkins_home/workspace/bbb-pipeline/backend
gradle clean build -x test

echo "*****"
echo "***Building DOCKER CONTAINERS***"
echo "*****"

# docker container 빌드
cd /var/jenkins_home/workspace/bbb-pipeline/pipeline
docker compose build --no-cache
```

```
# deploy.sh

#!/bin/bash

echo "*****"
echo "***DEPLOYING DOCKER CONTAINERS***"
echo "*****"

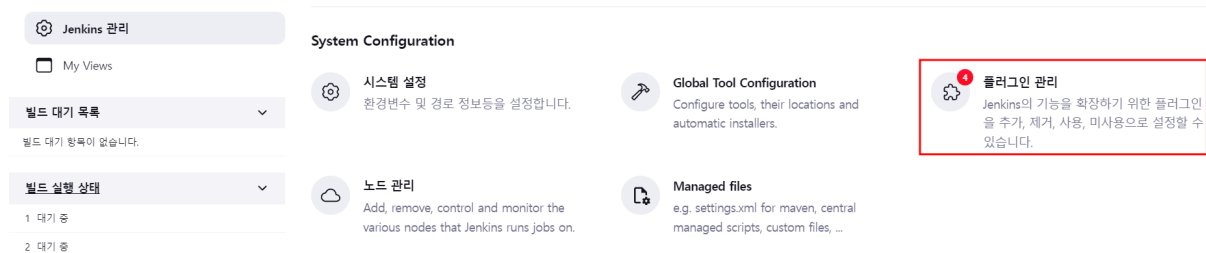
# docker container 실행
cd /var/jenkins_home/workspace/bbb-pipeline/pipeline
docker compose up -d

# dangling images 삭제
docker image prune -f
```

## 6.5. Jenkins 설정

### 6.5.1. Jenkins Plugin 추가설치

Jenkins 홈 화면 → “Jenkins 관리” → “플러그인 관리” → “Available plugins” 클릭 → 플러그인 검색 및 설치



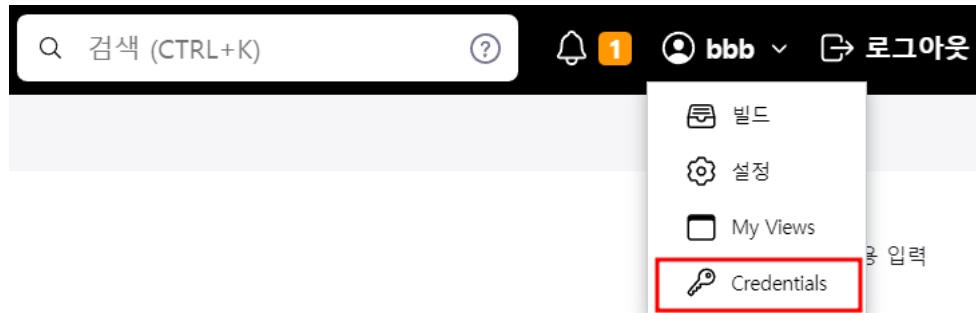
#### 필수 플러그인

- GitLab
- Gradle

- NodeJS
- Pipeline

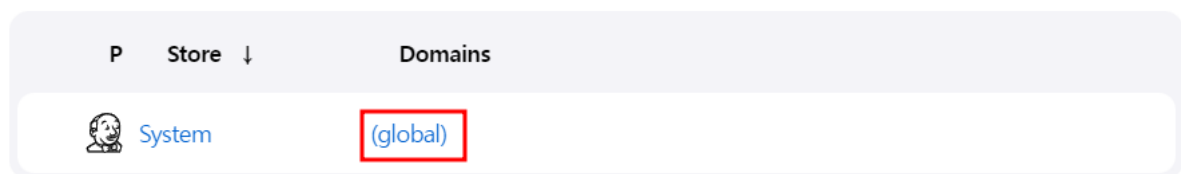
## 6.5.2. GitLab Credentials 설정

1. 아이디 → “Credentials” 클릭



2. “Stores scoped to Jenkins” → “(global)” → “+ Add Credentials” 클릭

### Stores scoped to Jenkins



아이콘: S M L

### Global credentials (unrestricted)



3. “Kind”에 “Username with password” 입력 → “Username”에 GitLab ID 입력 → “Password”에 Gitlab password 입력 → “ID”에 임의 아이디 입력 → 생성

#### New credentials

Kind

Username with password

Scope ?

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

Username ?

Treat username as secret ?

Password ?

ID ?

### 6.5.3. Jenkins Item 생성

1. “새로운 Item” 클릭
2. “Enter an item name”에 임의 Item 이름 입력 → “Pipeline” 클릭

#### Enter an item name

test-item

» Required field



#### Freestyle project

이것은 Jenkins의 주요 기능입니다. Jenkins은 어느 빌드 시스템과 어떤 SCM(형상관리)으로 묶인 당신의 프로젝트를 빌드할 것이고, 소프트웨어 빌드보다 다른 어떤 것에 자주 사용될 수 있습니다.



#### Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

3. “General” → “Do not allow concurrent builds” 클릭  
(한 빌드를 진행중이면 동시에 빌드를 진행하지 않게 한다)



Do not allow concurrent builds



Abort previous builds ?

4. “Build Triggers” → “Build when a change is pushed to GitLab” 클릭  
(WebHook 설정 : GitLab 특정 브랜치 push 시 자동 빌드 + 배포 설정)  
(해당 URL 복사 → WebHook 설정 시 사용 예정)



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

Never



5. “Build when a change is pushed to GitLab” 하위의 “고급...” 클릭



Comment (regex) for triggering a build ?

Jenkins please retry a build



6. "Secret token"의 "Generate" 클릭 후 생성된 토큰값 복사

Secret token ?

[Redacted Secret Token]

Generate

7. "Pipeline" → "Definition"에 Pipeline script from SCM 설정 → "SCM"에 "Git" 설정 → "Repository URL"에 프로젝트 GitLab URL 입력 → "Credentials"에 사전에 추가한 Credentials 입력

## Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

[Redacted Repository URL]

Credentials ?

[Redacted Credentials]

+ Add

고급...

8. "Branch Specifier"에 빌드 할 브랜치명 입력 (master일 시 \*/master)

Branches to build ?

Branch Specifier (blank for 'any') ?

refs/heads/develop

Add Branch

9. "Script Path"에 Jenkinsfile 경로 입력 → "Lightweight checkout" 해제 → 저장

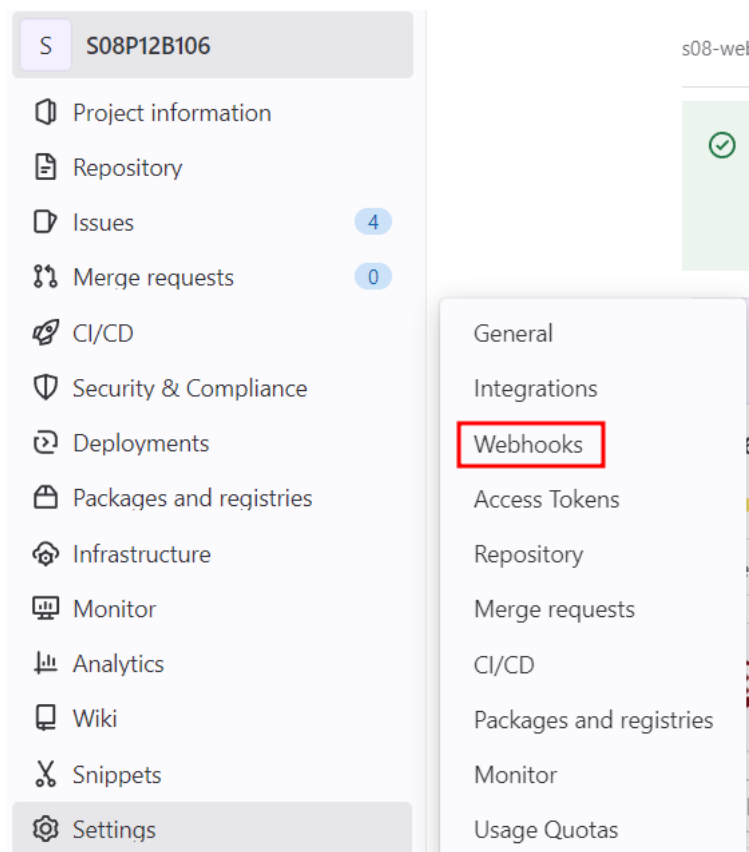
Script Path ?

pipeline/Jenkinsfile

☐ Lightweight checkout ?

#### 6.5.4. GitLab Webhook 설정

1. 프로젝트 GitLab → "Settings" → "Webhooks" 클릭



2. "URL"에 사전에 복사해놓은 Jenkins URL 입력 → "Secret token"에 사전에 복사해놓은 Secret token 입력 → "Push events" 클릭 후 WebHook 적용 브랜치 입력 (Jenkins Branch Specifier과 일치하여야 함)

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

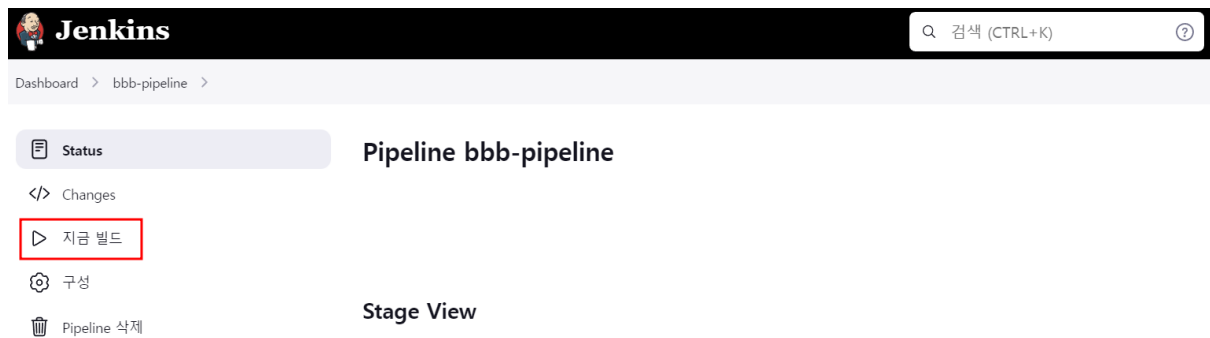
master

Push to the repository.

## 6.5.5. 빌드 및 배포

Option 1. 상기 WebHook 설정한 브랜치로 푸시

Option 2. Jenkins 홈 화면 → Jenkins Item 클릭 → “지금 빌드” 클릭



## 7. 외부서비스

### 7.1. 소셜 로그인 - Kakao

Kakao Developers REST API (카카오 로그인)

✓ OAuth 기반 소셜 로그인 API 제공

<https://developers.kakao.com/docs/latest/ko/kakaologin/rest-api>

#### 7.1.1. 애플리케이션 생성

1. Kakao developer에서 로그인 후 애플리케이션 추가
2. 카카오 로그인을 활성화

카카오 로그인 ON

### 활성화 설정

상태

ON

카카오 로그인 API를 활용하면 사용자들이 번  
상태가 OFF일 때도 카카오 로그인 설정 항목들  
상태가 ON일 때만 실제 서비스에서 카카오 로

### 3. 사이트 도메인, Redirect URI 등록

## Web

사이트 도메인

- 카카오 로그인 사용 시 Redire

## Redirect URI

Redirect URI

- 카카오 로그인에서 사용할 O

### 7.1.2. 키 생성, 동의 항목 선택

1. 앱 키 메뉴에서 REST API 키를 가져와 git에 올리지 않을 yaml 파일 작성
2. 카카오 로그인 동의 항목 설정

### 7.1.3. application-oauth.yml 작성

1. 외부로 노출 안 되도록 application-oauth.yml파일에 키와 관련 내용 작성

```
security:
  oauth2:
    client:
      registration:
        kakao:
          client-id: [REDACTED]
          client-secret: [REDACTED]
          client-name: [REDACTED]
          authorization-grant-type: [REDACTED]
          redirect-uri: [REDACTED]
          client-authentication-method: [REDACTED]
      sso: [REDACTED]
    provider:
      kakao:
        authorization-uri: [REDACTED]
        token-uri: [REDACTED]
        user-info-uri: [REDACTED]
        user-name-attribute: [REDACTED]
```

### 7.1.4. 카카오투터 사용자 정보 얻어오기

1. 카카오투터가 보내준 **인가코드** 를 Front에서 Back으로 전달

2. Back에서 카카오에게 인가코드를 전달하고 `access-token` 받아오기
3. `access-token`으로 카카오에 저장되어 있는 `user` 정보를 받아오기

## 7.2. 통합 수어 정보

### | 문화체육관광부 - 문화 공공 데이터 광장

- ✓ 국립국어원의 통합 수어정보 제공

<https://www.data.go.kr/data/15105243/openapi.do>

## 7.3. 클라우드 관계형 데이터베이스

### | AWS RDS

- ✓ AWS에서 제공하는 관계형 DB로 서비스 유저, 단어, 게임 관련 데이터를 저장한다

<https://aws.amazon.com/ko/rds/>