



포팅 매뉴얼 : Book! 빠지다

| SSAFY 9기 공통 프로젝트 부울경 2반 8팀, Reboot

0. 목차

0. 목차

I. 개발 환경

1. 프로젝트 기술 스택

2. 환경변수 설정하기

Frontend : .env

Backend : application.yml 등

3. 설정 파일

Nginx : nginx.conf

Jenkins : Jenkinsfile

docker-compose.yml

Backend : Dockerfile

Frontend : Dockerfile

4. 계정 정보 및 덤프 파일

→ 로컬 MySQL 접속 계정 정보

→ EC2 MySQL 접속 계정 정보

→ Jenkins 접속 계정 정보

→ Book! 빠지다 접속 계정 정보

II. 프로젝트 local 실행 가이드

1. Backend 실행 가이드

(1) IntelliJ 2023.1.3 설치하기

(2) Java 8 1.8.0_192 설치

(3) MySQL 8.0.33 설치

(4) Project Git Clone

(5) Springboot Application 실행시키기

2. Frontend 실행 가이드

(1) Visual Studio Code(VSC) 설치

(2) Project Git Clone

(3) Node.js 18.16.1 설치

(4) yarn 1.22.19 설치

(5) 패키지 설치

(6) 프로젝트 실행

III. 배포 방법 및 배포 실행 가이드

1. 방화벽 설치

EC2 필요한 프로그램 설치

2. Docker 및 docker-compose 설치

3. 도커 실행

4. 젠킨스 설치 및 실행

5. 젠킨스 초기 설정

6. Jenkins 플러그인 추가 설치

7. 젠킨스 Credential등록

8. 젠킨스 세부 설정

9. Gitlab WebHook설정

10. Jenkins 설정 후 서버 빌드 및 배포 방법

11. MySQL 환경설정

MySQL 원격접속 후 Database 생성 및 데이터 삽입

IV. 외부 서비스

1. Kakao OAuth2.0(카카오 로그인)

- (1) <https://developers.kakao.com/> 접속
- (2) 로그인 후, 내 애플리케이션 > 애플리케이션 추가하기
- (3) 전체 애플리케이션 목록 > 앱 설정 > 요약정보
- (4) 내 애플리케이션 > 앱 설정 > 플랫폼 > Web
- (5) 내 애플리케이션 > 제품 설정 > 카카오 로그인
- (6) 내 애플리케이션 > 고급 설정 > 허용 IP 주소
- (7) 내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목
- (7-1) 내 애플리케이션 > 앱 설정 > 비즈니스

I. 개발 환경

1. 프로젝트 기술 스택

- Frontend
 - Visual Studio Code(IDE) : 1.81.1
 - HTML5, CSS3, Javascript(ES6)
 - React : 18.2.0
 - Node.js : 18.16.1
 - yarn : 1.22.19
 - Recoil : 0.7.7
 - Recoil-perisist : 5.1.0
- Backend
 - IntelliJ(IDE) : 2023.1.3
 - Java : 1.8.0_192
 - SpringBoot : 2.7.14
 - Gradle
 - ORM : JPA(Hibernate)
 - Spring Security
 - java-jwt : 4.2.1
 - Swagger : 2.9.2

- CI/CD
 - AWS EC2
 - Nginx : 1.18.0
 - Ubuntu : Ubuntu 20.04 LTS
 - Docker : 24.0.5
 - Jenkins : 2.401.3
- Database
 - MySQL : 8.0.33
- Cooperation
 - Notion
 - MatterMost
 - Discord, Webax
 - Figma
 - Git-Flow
 - Github

2. 환경변수 설정하기

Frontend : .env

- 환경변수 설정 위치

```
frontend
└─ .env
```

- .env (Frontend 프로젝트 설정)

```
REACT_APP_API_URL = http://localhost:8080
```

Backend : application.yml 등

- 환경변수 설정 위치

```
backend
└─ src
   └─ main
      └─ resources
         ├── application.yml
         ├── application-local.yml
         ├── application-prod.yml
         └─ reboot-623ba-firebase-adminsdk-zj8bg-d7fdb28b4.json
```

- application.yml (Backend 프로젝트 설정)

```

# MySQL 설정

into-book:
  api:
    key: ttbjmj25885471405001

spring:
  profiles:
    active: local

# profiles:
#   include: oauth

jpa:
  database: mysql
  show-sql: true
  hibernate:
    ddl-auto: update
  properties:
    hibernate:
      show_sql: true
      format_sql: true
mvc:
  pathmatch:
    matching-strategy: ant_path_matcher

# createDatabaseIfNotExist: 데이터베이스가 존재하지 않으면 자동으로 생성
# useUnicode: 유니코드 사용 여부 설정
# characterEncoding: 문자열 인코딩 종류 설정
# characterSetResult: 결과값의 인코딩 종류 설정
# useSSL: SSL 사용여부 설정

# spring.jpa.properties.hibernate.show_sql : 하이버네이트가 실행한 모든 SQL문을 콘솔로 출력
# spring.jpa.properties.hibernate.format_sql : SQL문을 가독성 있게 표현

swagger:
  project:
    base-package: com.reboot.intobook.controller.api

jwt:
  secretKey: sc136as1c5qw4c13ax1cz3x5c4as3c1a35scczx3casc54;

access:
  expiration: 3600000000 # 100시간(60분) (1000L(ms -> s) * 60L(s -> m) * 60L(m -> h))
  header: Authorization

refresh:
  expiration: 12096000000 # (1000L(ms -> s) * 60L(s -> m) * 60L(m -> h) * 24L(h -> 하루) * 14(2주))
  header: Authorization-refresh

logging:
  level:
    org:
      hibernate:
        type:
          descriptor:
            sql: trace

fcm:
  certification: src/main/resources/reboot-623ba-firebase-adminsdk-zj8bg-d7fdb28b4.json

```

- **application-local.yml** (local 설정)

```

server:
  address: localhost
  port: 8080

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/intobook?createDatabaseIfNotExist=true&useUnicode=true&characterEncoding=utf8
    username: root
    password: root

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: d65bd1ef2f9d2f8056ef153efcc983e7
            client-secret: XJnJnScvebxqQNWetoHdUMI9x17WtLn
            redirect-uri: http://localhost:8080/login/oauth2/code/kakao
            client-authentication-method: POST
            authorization-grant-type: authorization_code
            scope: profile_nickname, account_email
            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

preEndPoint: ""

redirect:
  url: http://localhost:3000

```

- **application-prod.yml (배포 설정)**

```

server:
  address: 0.0.0.0
  port: 8080
  servlet:
    context-path: /api
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://3.38.116.69:3306/intobook?createDatabaseIfNotExist=true&useUnicode=true&characterEncoding=utf8
    username: root
    password: root

  security:
    oauth2:
      client:
        registration:
          kakao:
            client-id: d65bd1ef2f9d2f8056ef153efcc983e7
            client-secret: XJnJnScvebxqQNWetoHdUMI9x17WtLn
            redirect-uri: https://intobook.kro.kr/api/login/oauth2/code/kakao
            client-authentication-method: POST
            authorization-grant-type: authorization_code
            scope: profile_nickname, account_email
            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
preEndPoint: "/api"

redirect:
    url: https://intobook.kro.kr

```

- reboot-623ba-firebase-adminsdk-zj8bg-d7fdb428b4.json

```

{
  "type": "service_account",
  "project_id": "reboot-623ba",
  "private_key_id": "d7fdb428b42ecccf3d861c182e50202e96f4200f",
  "private_key": "-----BEGIN PRIVATE KEY-----\nMIIEVQIBADANBgkqhkiG9w0BAQEFAASCBCwggSjAgEAAoIBAQC6mAiophaQ
  "client_email": "firebase-adminsdk-zj8bg@reboot-623ba.iam.gserviceaccount.com",
  "client_id": "112500500026491853713",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/firebase-adminsdk-zj8bg%40rebo
  "universe_domain": "googleapis.com"
}

```

3. 설정 파일

Nginx : nginx.conf

- 설정 파일 위치

```
/etc/nginx/sites-enabled/default
```

- nginx.conf

```

server {
    listen 80 default_server;
    listen [::]:80 default_server;

    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    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 intobook.kro.kr www.intobook.kro.kr;
    ssl_certificate /home/ubuntu/certificate.pem;
    ssl_certificate_key /home/ubuntu/private.key;
    ssl_prefer_server_ciphers on;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
    }

    location /api {
        proxy_pass http://localhost:8080;
        proxy_set_header X-Real_IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    }
}

```

```

        proxy_set_header Host $http_host;
    }

    location /ws {
        proxy_pass https://intobook.kro.kr;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "Upgrade";
    }

}

server {
    listen 80;
    server_name intobook.kro.kr i9e208.p.ssafy.io;
    return 301 https://intobook.kro.kr$request_uri;
}

```

Jenkins : Jenkinsfile

- 설정 파일 위치

```
/home/ubuntu/compose/jenkins-dockerfile/Dockerfile
```

- Jenkinsfile

```

FROM jenkins/jenkins:lts

USER root
RUN apt-get update &&\
    apt-get upgrade -y &&\
    apt-get install -y openssh-client

```

docker-compose.yml

- 설정 파일 위치

```
/home/ubuntu/compose/docker-compose.yml
```

- docker-compose.yml

```

version: "3"
services:
  jenkins:
    container_name: jenkins-compose
    build:
      context: jenkins-dockerfile
      dockerfile: Dockerfile
    user: root
    ports:
      - 8000:8080
      - 8888:50000
    volumes:
      - /home/ubuntu/compose/jenkins:/var/jenkins_home
      - /home/ubuntu/compose/.ssh:/root/.ssh
  spring:
    container_name: spring-compose
    build:

```

```

        context: spring-dockerfile
        dockerfile: Dockerfile
    ports:
        - 8080:8080
    volumes:
        - /home/ubuntu/compose/jenkins/workspace/intobook/intobook/build/libs:/deploy
        - /home/ubuntu/compose/jenkins/workspace/intobook/intobook/src/main/resources/reboot-623ba-firebase-adminsdk-zj8bg-d7fdb28b4.json:/app/fcm.json
    react:
        container_name: react-compose
        build:
            context: /home/ubuntu/compose/jenkins/workspace/intobook_front/client/intobook
            dockerfile: Dockerfile
        ports:
            - 3000:3000
        volumes:
            - /home/ubuntu/compose/jenkins/workspace/intobook_front/client/intobook:/app

```

Backend : Dockerfile

- 설정 파일 위치

```
/home/ubuntu/compose/spring-dockerfile/Dockerfile
```

- Dockerfile

```

FROM openjdk:8-jdk
ENTRYPOINT java -jar /deploy/intobook-0.0.1-SNAPSHOT.jar
ENV SPRING_PROFILES_ACTIVE=prod
ENV FCM_CERTIFICATION=/app/fcm.json
EXPOSE 8080

```

Frontend : Dockerfile

- 설정 파일 위치

```
/home/ubuntu/compose/jenkins/workspace/intobook_front/client/intobook/Dockerfile
```

- Dockerfile

```

FROM node:latest
WORKDIR /app
COPY . .
RUN yarn build
EXPOSE 3000
CMD ["yarn", "start"]

```

4. 계정 정보 및 덤프 파일

→ 로컬 MySQL 접속 계정 정보

- ID : root
- PW : root

→ EC2 MySQL 접속 계정 정보

- ID : root
- PW : root
- RDS 덤프 파일

backup_intobook.zip

➡ Jenkins 접속 계정 정보

- ID : reboot
- PW : reboot123
- Jenkins URL : <http://i9e208.p.ssafy.io:8000>

➡ **Book! 빠지다 접속 계정 정보**


- ID : reboot208
- PW : test*123
- swagger
 - 접속 URL : <https://intobook.kro.kr/api/swagger-ui.html>
 - 토큰(로그인할 때, network 콘솔에서 확인가능) : Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJBcDZlbiIsImV4cCI6MTY5MjYyXkWTJT57o4gKSMym6MOo9-1KiPoYc1cj-oU5LIPa5jn5j1ozaZHReB0x-jy5YEj3KXkjWNCZUgrK3Niq6Y2xg

II. 프로젝트 local 실행 가이드

1. Backend 실행 가이드


(1) IntelliJ 2023.1.3 설치하기

- 이전 버전 설치




IntelliJ IDEA

IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains



IntelliJ IDEA

The Leading Java and Kotlin IDE




<https://www.jetbrains.com/idea/download/other.html>

- 인코딩 설정 UTF-8로 통일시키기

(2) Java 8 1.8.0_192 설치

- jdk 버전8

 https://www.java.com/download/ie_manual.jsp

- **java 환경변수 설정하기**

JDK 8 다운로드 및 설치하기, 환경변수 설정 [Java개발환경 구축하기 1]

JDK8 설치하기 (윈도우10) jdk는 지금 14버전까지 나와있다. 근데 왜 우리는 왜 8버전을 쓸까?? 그러게여.. 알려주세요.. 아마 9버전 이상부터는 상업적 이용을 위해선 돈을 지불하고 사용해야 하기 때문일 것이다. 그래도 양심은 있는지 무료로 사용 가능한 8버전은 현

<https://the-duchi.tistory.com/4>



(3) MySQL 8.0.33 설치

- 설치 가이드

MySQL 다운로드 및 설치하기(MySQL Community 8.0)

SQL을 본격적으로 사용하려면 DBMS를 설치해야 합니다. 여러 가지 DBMS 중에서 MySQL 설치 하는 방법을 알아보고, 정상적으로 설치가 되었는지 확인하는 방법을 알아 보겠습니다. 2021년 10월 기준 MySQL Community 8.0.21...

<https://hongong.hanbit.co.kr/mysql-다운로드-및-설치하기mysql-community-8-0/>

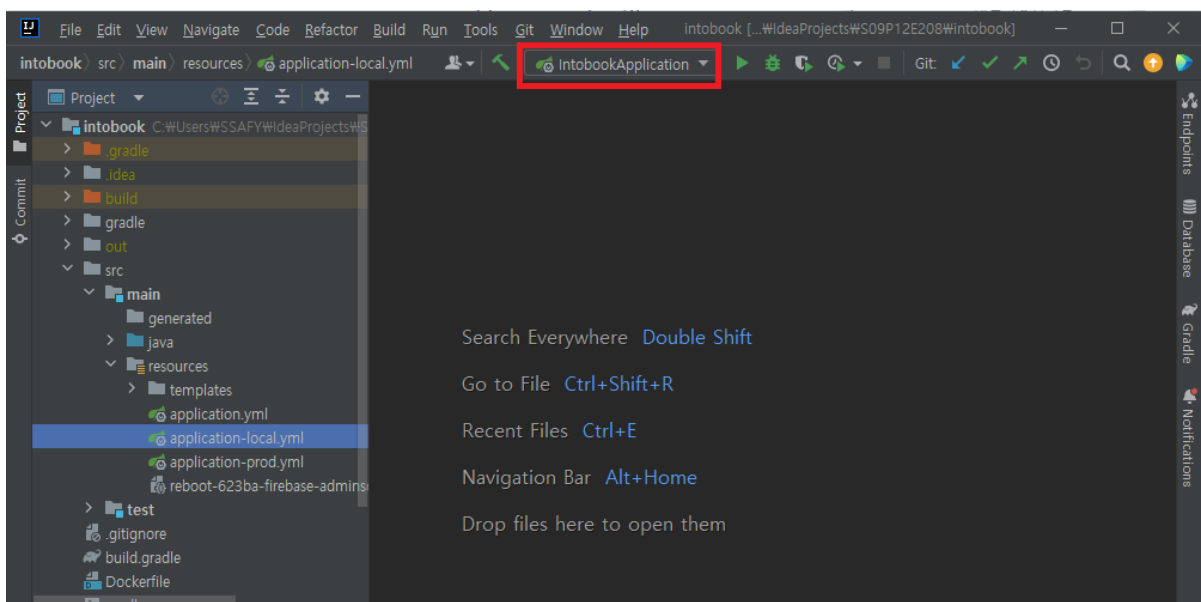


- 로컬 사용자 계정 (ID: root, PW: root) 로 설정하기 → `application-local.yml` 에 명시

(4) Project Git Clone

```
> git clone https://lab.ssafy.com/s09-webmobile2-sub2/S09P12E208.git # git repo 클론하기
> cd S09P12E208 # 프로젝트가 클론된 폴더로 이동
```

(5) Springboot Application 실행시킴



2. Frontend 실행 가이드

(1) Visual Studio Code(VSC) 설치

<https://code.visualstudio.com/download>


(2) Project Git Clone

```
> git clone https://lab.ssafy.com/s09-webmobile2-sub2/S09P12E208.git # git repo 클론하기
> cd S09P12E208 # 프로젝트가 클론된 폴더로 이동
```

(3) Node.js 18.16.1 설치

Node.js

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

 <https://nodejs.org/ko>



(4) yarn 1.22.19 설치

```
> npm install --global yarn # --global을 이용해 전역에 yarn 설치하기
```

(5) 패키지 설치

```
> yarn i # package.json에 연결된 라이브러리를 모두 한 번에 다운로드하기
```

(6) 프로젝트 실행

```
> yarn start # 프로젝트 시작
```

III. 배포 방법 및 배포 실행 가이드

1. 방화벽 설치

- ssh 포트 허용

```
ssh -i {pemkey.pem} ubuntu@{서버 도메인}
$ sudo ufw allow 22/tcp # ssh는 tcp 프로토콜만 허용하는게 맞음
$ sudo ufw status verbose # 방화벽 포트
$ sudo ufw deny 22/tcp
$ sudo ufw delete deny 22/tcp
```

EC2 필요한 프로그램 설치

1. git 설치

```
ubuntu@ip-172-26-10-240:/etc/apt$ sudo apt-get install git
ubuntu@ip-172-26-10-240:/etc/apt$ git --version
```

2. docker 설치

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

패키지들이 시스템에 설치되어, HTTPS를 통한 패키지 다운로드, 인증서 관리, 데이터 전송, 암호화 및 디지털 서명 처리, 소프트웨어 저장소 관리 등의 기능을 사용할 수 있게 됨

- **apt-transport-https** : 이 패키지는 APT(Advanced Package Tool) 패키지 관리 시스템에서 HTTPS를 통한 패키지 다운로드를 지원합니다. 이를 통해 암호화된 연결을 사용하여 패키지를 다운로드할 수 있습니다.
- **ca-certificates** : 이 패키지는 일반적으로 사용되는 CA(인증 기관) 인증서를 제공합니다. 이 인증서들은 SSL/TLS 암호화 및 인증에 사용되며, 시스템이 신뢰할 수 있는 인증서를 사용하도록 합니다.
- **curl** : 이 도구는 명령행에서 데이터 전송을 수행할 수 있도록 합니다. 여러 프로토콜을 지원하며, URL 문법을 사용하여 데이터를 전송하거나 받을 수 있습니다.
- **gnupg-agent** : 이 패키지는 GnuPG 암호화 및 디지털 서명 도구를 사용하는 데 필요한 프로세스를 실행합니다. 이 에이전트는 암호화 키를 관리하고, 암호화 및 디지털 서명과 관련된 동작을 처리합니다.
- **software-properties-common** : 이 패키지는 소프트웨어 저장소 및 관련 설정을 관리하기 위한 도구를 제공합니다. 예를 들어, 외부 저장소를 추가하거나 PPA(Personal Package Archive)를 사용할 때 이 패키지가 필요합니다.

3. Docker의 GPG key 인증

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Docker Repository등록

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

5. 도커설치

```
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. docker-compose 설치

```
sudo curl -L \
"https://github.com/docker/compose/releases/download/1.28.5/docker-compose-$(uname -s)-$(uname -m)" \
-o /usr/local/bin/docker-compose
# $(uname -s) : 현재 시스템의 운영체제, 현재 운영체제에 맞게 이름을 동적으로 불러와서 요청을 보냄
```

```
# "$(uname -m)" : 현재 시스템의 아키텍처, x86 아키텍처의 경우 "$(uname -m)"은 "x86_64"를 반환
# 이 명령어는 외부에서 그대로 파일을 가져와서 현재의 시스템에 올려 놓는 것이다.
# 결과적으로 "/usr/local/bin/" 경로에 "docker-compose" 라는 이름의 파일로 다운로드.
# 참고) https://github.com/docker/compose/releases 에서 최신 버전 확인이 가능하다.
# 파일별 저장 url을 확인 후 작성하여야 합니다.
sudo chmod +x /usr/local/bin/docker-compose # chmod 를 통해서 실행이 가능하게 세팅
docker-compose -v # docker-compose 명령이 제대로 먹히는 지 확인한다.
```

7. 도커 실행

```
sudo systemctl enable docker : 시스템 부팅 시 Docker를 자동으로 실행하도록 설정
sudo service docker start : Docker 서비스를 수동으로 시작
```

8. gradle 설치

```
sudo apt update
sudo apt install openjdk-11-jdk
# Gradle 다운로드 및 설치
wget https://services.gradle.org/distributions/gradle-7.6.1-bin.zip -P /tmp
sudo unzip -d /opt/gradle /tmp/gradle-*.zip
# Gradle 7.6.1을 다운로드하여 "/tmp" 디렉토리에 저장한 다음, "/opt/gradle" 디렉토리에 압축을 해제합니다.
# 환경 변수 설정
export GRADLE_HOME=/opt/gradle/gradle-7.6.1
export PATH=${GRADLE_HOME}/bin:${PATH}
source /etc/profile.d/gradle.sh # 적용
gradle -v # 확인
sudo chmod +x gradlew # 권한 부여
```

9. 프로젝트 클론 받기

10. 필요한 이미지 저장

11. 프로젝트 빌드

12. 도커 컴포즈로 올리기

13. 원격 데이터베이스 생성하기

```
ubuntu@ip-172-26-3-38:~/S08P22D208/Server$ sudo docker ps
CONTAINER ID IMAGE COMMAND CREATED
45b1db8c07d1 server_module-batch "java -Duser.timezone..." 3 minutes ago
5034bf51ff93 mariadb "docker-entrypoint.s..." 3 minutes ago
51570b204727 redis:alpine "docker-entrypoint.s..." 3 minutes ago
ubuntu@ip-172-26-3-38:~/S08P22D208/Server$ sudo docker exec -it 5034bf51ff93 /
root@5034bf51ff93:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 9
Server version: 10.11.2-MariaDB-1:10.11.2+maria-ubu2204 mariadb.org binary dis
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
MariaDB [(none)]> create database coredb;
Query OK, 1 row affected (0.000 sec)
MariaDB [(none)]> create database batchdb;
Query OK, 1 row affected (0.000 sec)
MariaDB [(none)]> exit
```

2. Docker 및 docker-compose 설치

1. 패키지 업데이트 진행

```
sudo apt update & apt upgrade
```

2. Docker 설치에 필요한 필수 패키지 설치

```
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-properties-common
```

3. Docker의 GPG key 인증

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Docker Repository 등록

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
```

5. Docker 설치

```
sudo apt-get update && sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. docker-compose 설치

```
sudo curl -L \
"https://github.com/docker/compose/releases/download/1.28.5/dockercompose-${uname -s}-${uname -m}" \
-o /usr/local/bin/docker-compose
# 이 명령어는 외부에서 그대로 파일을 가져와서 현재의 시스템에 올려 놓는 것이다.
# 결과적으로 "/usr/local/bin/" 경로에 "docker-compose" 라는 이름의 파일로 다운로드.
# 참고) https://github.com/docker/compose/releases 에서 최신 버전 확인이 가능하다.
# 최신 버전을 설치하고 싶다면 위 명령어에 보이는 1.28.5 라는 버전 숫자를 바꿔주면 된다!
sudo chmod +x /usr/local/bin/docker-compose # chmod 를 통해서 실행이 가능하게 세팅
docker-compose -v # docker-compose 명령이 제대로 먹히는 지 확인한다.
```

3. 도커 실행

```
sudo systemctl enable docker
sudo service docker start
```

4. 젠킨스 설치 및 실행

1. **Dockerfile**을 만들어서 jenkins 이미지를 활용해 docker.sock을 활용해 통신가능하도록 컨테이너를 띄워준다.

```
FROM jenkins/jenkins:lts
# Jenkins 공식 이미지를 기반으로하는 Docker 이미지를 생성
```

```

USER root
# Docker를 설치하려면 root 권한이 필요합니다. 따라서 사용자를 root로 변경
RUN apt-get update \
&& apt-get -y install lsb-release \
&& curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /usr/share/keyrings/docker-ar
&& echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.
&& apt-get update \
&& apt-get -y install docker-ce docker-ce-cli containerd.io
RUN usermod -u 1000 jenkins && \
groupmod -g 998 docker && \
usermod -aG docker jenkins
# 패키지 목록을 업데이트
# lsb-release 패키지 설치
# Docker GPG 키를 다운로드하고 /usr/share/keyrings/docker-archive-keyring.gpg 파일로 저장
# Docker 레포지토리를 /etc/apt/sources.list.d/docker.list 파일에 추가
# Docker CE, Docker CLI, containerd.io 패키지를 설치
# Jenkins 사용자의 UID를 1000으로 변경하고 Docker 그룹의 GID를 998로 변경
# Jenkins 사용자를 Docker 그룹에 추가

```

- 이때 주의할점은 아래의 옵션들이다. `usermod -u {호스트의사용자아이디}` `groupmod -g {호스트의 도커 그룹 아이디}`
- ubuntu host에서 호스트의 사용자 아이디를 가져오려면 `id -u` 명령어를 활용하고, 도커 그룹 아이디를 가져오고 싶다면 `cat/etc/group | grep docker` 를 사용하면 된다.

2. 이미지를 만든다

```
docker build -t my-jenkins:0.1 .
```

3. 이미지 확인

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
my-jenkins	0.1	d37227db069f	32 minutes ago	985MB

4. volume 만들기

```
docker volume create jenkins
```

5. volume 확인 (docker volume ls)

DRIVER	VOLUME NAME
local	jenkins

6. jenkins 컨테이너 실행

```

docker run -d --name jenkins \
-v /var/run/docker.sock:/var/run/docker.sock \
-v jenkins:/var/jenkins_home \
-p 8080:8080 my-jenkins:0.1

/usr/local/bin/docker-compose

```

5. 젠킨스 초기 설정

1. **http://서버아이피:8080**으로 접속 한다.
2. 젠킨스에 처음 접속하면 초기 **관리자 계정의 비밀번호를 입력**하라고 나온다.

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

3. 우선 **jenkins 컨테이너에** 접속한다

```
docker exec -it jenkins /bin/bash
```

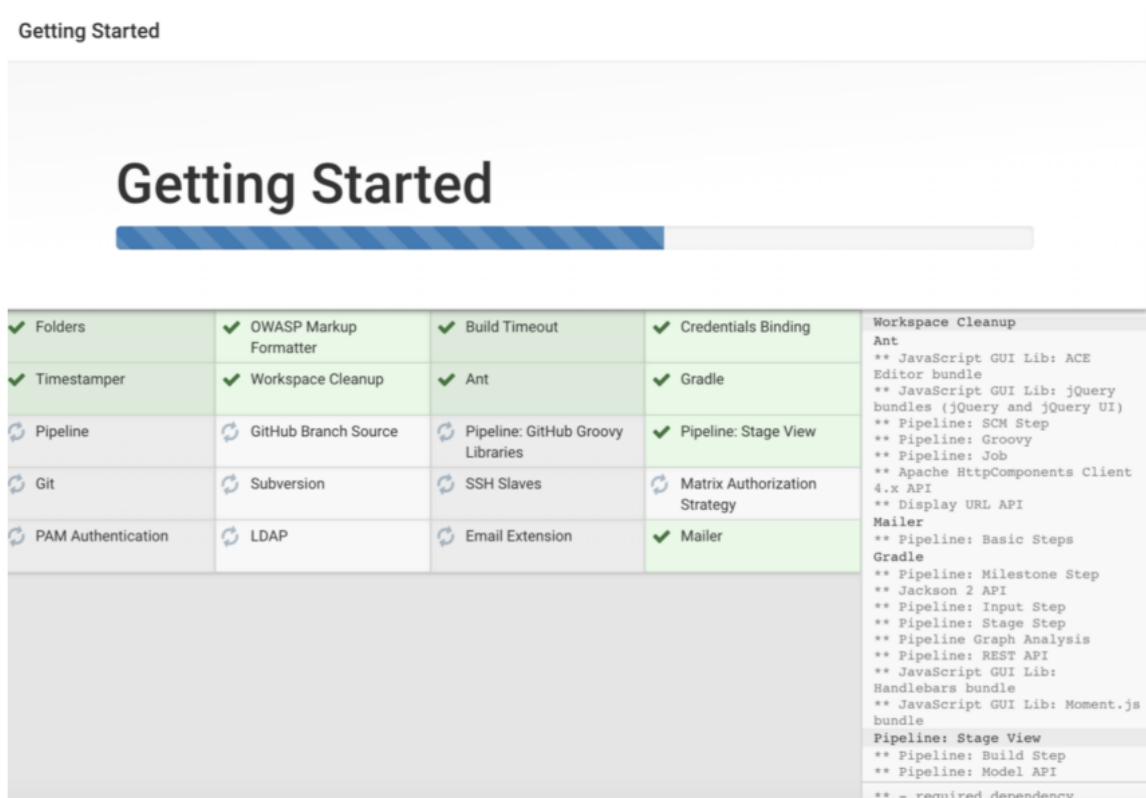
4. 초기 관리자 비밀번호를 확인한다

```
cat /var/jenkins_home/secrets/initialAdminPassword
```

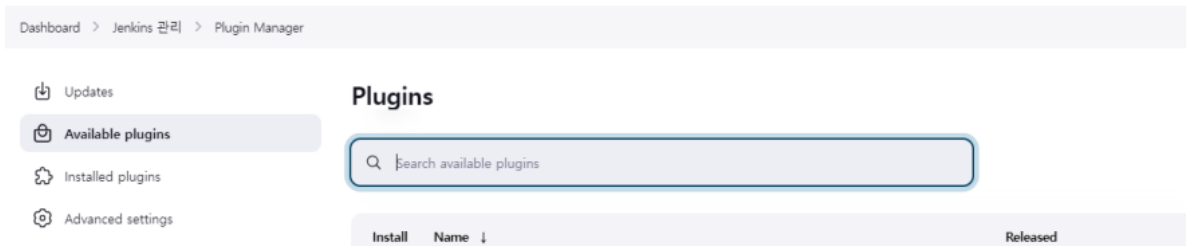
5. 비밀번호를 입력하면 아래와 같은 화면이 나오는데, **Install suggested plugins** 클릭



6. 그대로 쪽 설치진행 하면 아래와 같이 진행이 된다. 설치가 모두 완료되면 관리자의 아이디 패스워드 설정하는 창이 나오고 본인이 설정하고싶은 패스워드로 설정하면된다.



6. Jenkins 플러그인 추가 설치



설치할 플러그인 목록

- Gitlab
- Gradle(이미 설치되어 있다면 설치 안해줘도 됨)

플러그인 검색창에서 Gitlab, Gradle 검색 후 설치해준 뒤 jenkins를 재시작 시켜준다.

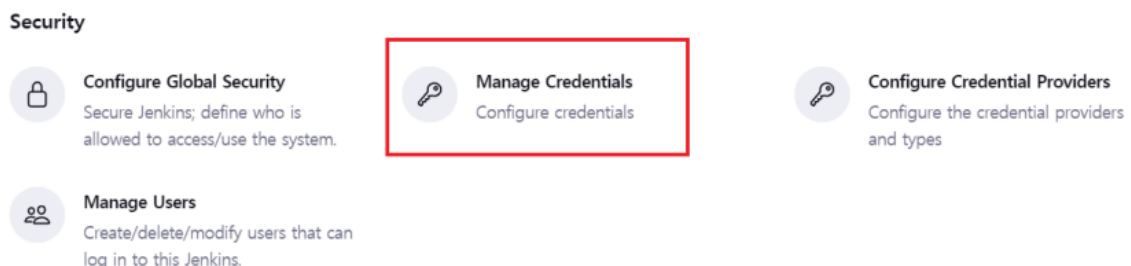
설치완료가 되면 jenkins admin페이지 하단에 restart jenkins라는 버튼이 생기고, 눌러주면 자동으로 재시작이 완료된다.

그러나 가끔 재시작이 정상적으로 되지 않는 경우가 생기는데 그때는 EC2에 원격접속 후 jenkins 컨테이너를 아래와 같은 명령어를사용해 재실행해준다.

```
docker restart {container_id 또는 container name}
```

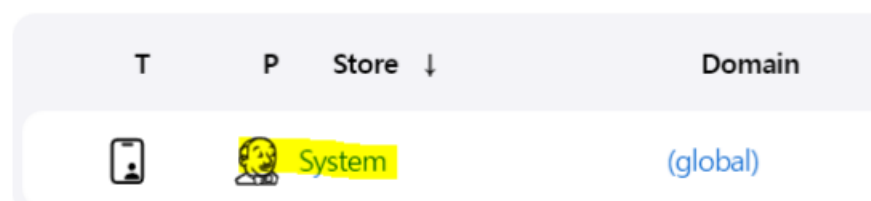
7. 젠킨스 Credential등록

1. Jenkins관리 → Manage Credential에 들어간다

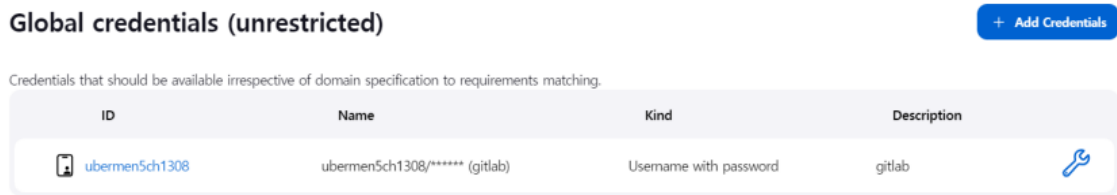


2. System 선택

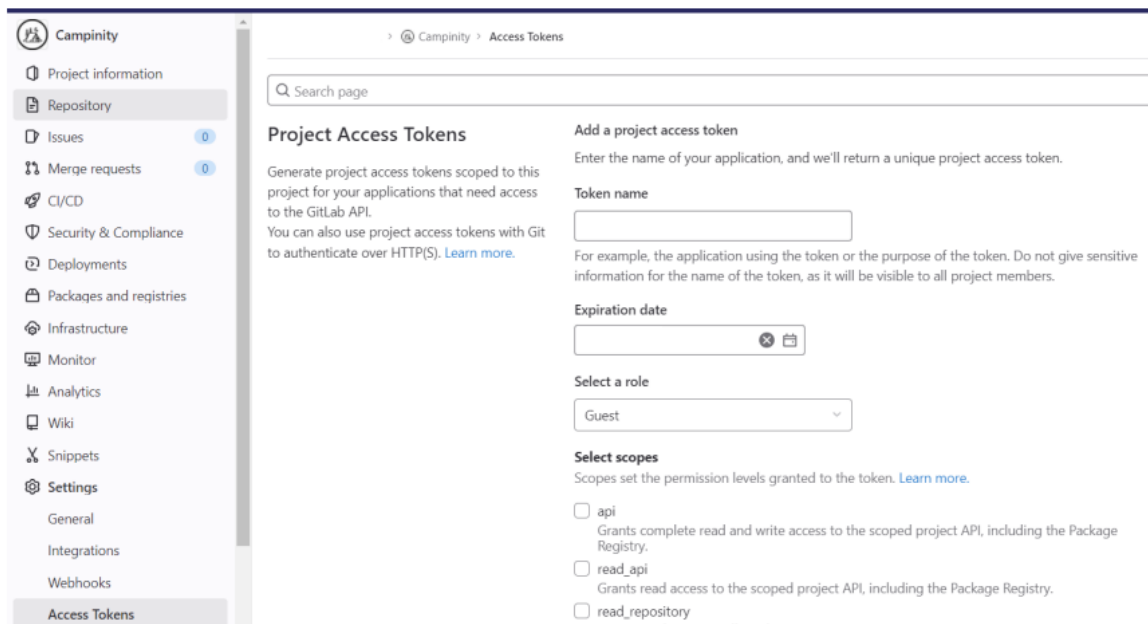
Credentials



3. 우측 상단에 **Add Credentials** 클릭



4. 우선 Gitlab 프로젝트 Repository에 들어가서 **accesstoken**을 발급받는다



5. 발급 받은 accessToken을 **Credential**을 만들때 **password**로 입력해주고 나머지 부가 정보들도 입력해주고 Create해준다

8. 젠킨스 세부 설정

1. 젠킨스 로그인을 완료하고 새로운 Item추가를 클릭한다.
2. item이름을 입력하고 **Freestyle project**를 선택한다.



3. 소스 코드 관리 목록중 **Git**을 선택하고 **Repository URL**에 **Gitlab프로젝트 URL**을 입력하고 이전에 설정해 둔 **Credential**로 선택해준다.

The screenshot shows the GitLab Configuration page. At the top, there's a breadcrumb trail: Dashboard > [blank] > Configuration. Below this is the 'Configure' section with a 'General' tab selected. The 'General' tab contains several options: '소스 코드 관리' (Source Code Management), '빌드 유발' (Build Trigger), '빌드 환경' (Build Environment), 'Build Steps', and '빌드 후 조치' (Build After Action). Below these options, there are radio buttons for 'None' and 'Git'. The 'Git' option is selected. Under the 'Git' option, there's a 'Repositories' section. This section contains a 'Repository URL' field, which is highlighted with a red box. Below the 'Repository URL' field, there's a red error message: 'Please enter Git repository.' Below the error message, there's a 'Credentials' section. This section contains a dropdown menu with the option '- none -' selected. Below the dropdown menu, there's a '+ Add' button, which is also highlighted with a red box. At the bottom of the 'Repositories' section, there's a '고급...' (Advanced...) button.

4. 바로 아래에 어떤 브랜치에서 **commit**들과와서 **Integration** 시킬지 **branch**를 명시해준다. (deploy-be로 설정)

Branches to build ?

Branch Specifier (blank for 'any') ?

*/deploy-be

Add Branch

5. 빌드 유발 설정 부분을 아래와 같이 설정해준다 .

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://i8d101.p.ssafy.io:8080/project/campinity-develop2 ?

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

6. Build Steps설정은 다음과 같다

Build Steps

≡ Invoke Gradle script ?
Help for feature: Invoke Gradle

● Invoke Gradle ?

Gradle Version
Gradle 7.6.1

☐ Use Gradle Wrapper ?

Tasks ?
build -p /var/jenkins_home/workspace/ Server -x test

고급 ▾

≡ Execute shell ?

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

```
cd Server
docker build --no-cache -t my-flask:0.1 ./flask-server
docker-compose up -d --build
if [ "$(docker images -f 'dangling=true' -q)" ]; then
  docker rmi $(docker images -f 'dangling=true' -q)
fi
```

고급 ^

☒ Enable [ci-skip]

☒ Ignore WIP Merge Requests

Labels that forces builds if they are added (comma-separated)

☒ Set build description to build cause (eg. Merge request or Git Push)

☐ Build on successful pipeline events

Pending build name for pipeline ?

☐ Cancel pending merge request builds on update

Allowed branches

☒ Allow all branches to trigger this job ?

☐ Filter branches by name ?

☐ Filter branches by regex ?

☐ Filter merge request by label

Secret token ?

Generate

9. Gitlab WebHook설정

1. **Gitlab WebHook** 탭에 들어가서 **설정값**들을 입력해준다.

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 `jenkins-server-url/jenkins-item-name`

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.

2. 설정값 중 **Secret token**은 jenkins서버에 접속해서 item(campinity-develop) → 구성 → 빌드유발 → 고급 → Secrettoken → generate 클릭해서 **토큰을 발급받고 입력**해준다.

☒ Build when a change is pushed to GitLab. GitLab webhook URL `http://[redacted]/project/[redacted]`

Enabled GitLab triggers

Push Events	<input checked="" type="checkbox"/>
Opened Merge Request Events	<input checked="" type="checkbox"/>
Accepted Merge Request Events	<input type="checkbox"/>
Closed Merge Request Events	<input type="checkbox"/>
Rebuild open Merge Requests	Never
Approved Merge Requests (EE-only)	<input type="checkbox"/>
Comments	<input checked="" type="checkbox"/>
Comment (regex) for triggering a build	Jenkins

Enable [ci-skip] ☒

Ignore WIP Merge Requests ☒

Set build description to build cause (eg. Merge request or Git Push) ☒

Build on successful pipeline events ☐

Pending build name for pipeline

Cancel pending merge request builds on update ☐

Allowed branches

☒ Allow all branches to trigger this job

☐ Filter branches by name

☐ Filter branches by regex

☐ Filter merge request by label

Secret token

Generate

3. Gitlab WebHook으로 다시 돌아가서 WebHook test

SSL verification

☒ Enable SSL verification

Save changes

Test ▾

Delete

Status	Trigger	Elapsed time	Request time	
200	Merge Request Hook	0.02 sec	just now	View details

10. Jenkins 설정 후 서버 빌드 및 배포 방법

1. 자동으로 빌드 및 배포

jenkins에 등록된 project URL에 Merge Request가 accept되면 자동으로 빌드 및 배포가 된다.

2. 수동으로 빌드 및 배포

지금 빌드 버튼을 누르면 item 설정된 URL 및 branch를 jenkins가 인식하고 commit들을 fetch후 build 및 deploy를 진행한다.

Dashboard > campinity-develop >

상태

</> 변경사항

작업공간

▶ 지금 빌드

⚙️ 구성

🗑️ Project 삭제

✎️ Rename

🔆 Build History

Project campinity-develop

고정링크

- Last build, (#225), 25 min 전
- Last stable build, (#225), 25 min 전
- Last successful build, (#225), 25 min 전
- Last failed build, (#131), 5 days 15 hr 전
- Last unsuccessful build, (#155), 3 days 3 hr 전
- Last completed build, (#225), 25 min 전

추이 ▾

참고 사항

위의 과정을 그대로 따라서 거쳐온다면, 서버가 제대로 작동하지 않을것이다. 그 이유는 MySQL 컨테이너에 Database가 생성되지않은 상태이기 때문이다. 따라서 EC2에 원격접속 후 **DB관련 환경설정**을 해주어야한다.

11. MySQL 환경설정

1. EC2 접속

```
ssh -i I9E208.pem ubuntu@i9e208.p.ssafy.io
```

2. MySQL container 접속

```
docker exec -it database /bin/bash
```

3. MySQL 서버에 root계정으로 접속

```
mysql -u root -p
```

4. User 등록

```
create user userid@접속할 host 외부아이피 identified by '비밀번호';
```

5. 권한 부여

```
GRANT ALL PRIVILEGES ON DB명.테이블 TO 계정아이디@host IDENTIFIED BY '비밀번호';
```

6. 권한 반영

```
flush privileges;
```

MySQL 원격접속 후 Database 생성 및 데이터 삽입

1. MySQL Workbench로 접속

MariaDB서버에서 root권한을 생성한 user의 username과 password를 설정해주고, hostname은 host의 아이피(Domain Name)를 입력해준다. 포트는 3306

Connection Name:

Connection Remote Management System Profile

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.

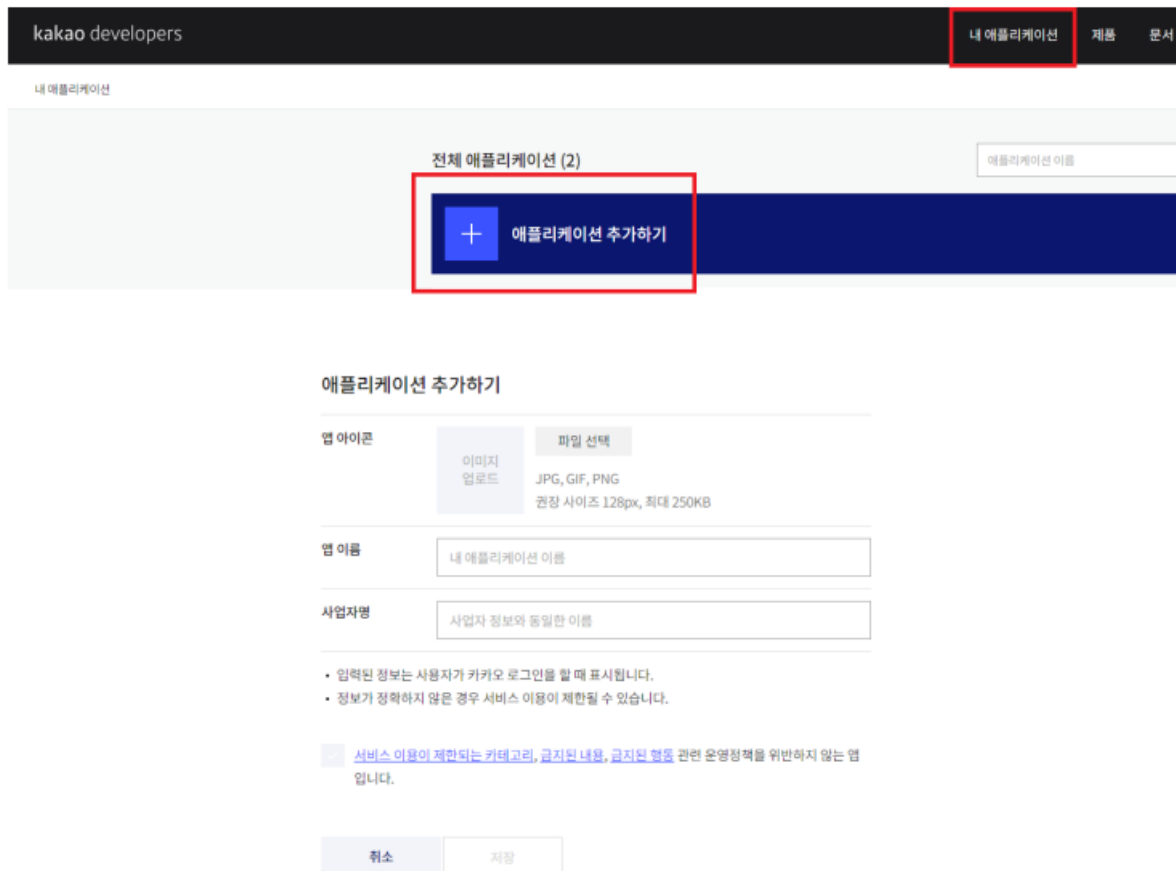
2. 접속 후 project repository/exec 폴더에있는 **backup.sql**을 다운받아서 **sql**을 **execute** 해준다

IV. 외부 서비스

1. Kakao OAuth2.0(카카오 로그인)

(1) <https://developers.kakao.com/> 접속

(2) 로그인 후, 내 애플리케이션 > 애플리케이션 추가하기



The image shows the 'Add Application' form on the Kakao Developers portal. The top navigation bar includes 'kakao developers', '내 애플리케이션' (highlighted with a red box), '계통', and '문서'. Below the navigation bar, the page title is '내 애플리케이션'. The main content area shows '전체 애플리케이션 (2)' and a button '+ 애플리케이션 추가하기' (highlighted with a red box). The form itself is titled '애플리케이션 추가하기' and contains the following fields:

- 앱 아이콘**: Includes an '이미지 업로드' button and a '파일 선택' button. Supported formats are JPG, GIF, PNG, with a maximum size of 128px by 128px and 250KB.
- 앱 이름**: A text input field with the placeholder '내 애플리케이션 이름'.
- 사업자명**: A text input field with the placeholder '사업자 정보와 동일한 이름'.

Below the form, there are two bullet points:

- 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.
- 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

A warning message in blue text states: '서비스 이용이 제한되는 카테고리, 금지된 내용, 금지된 행동 관련 운영정책을 위반하지 않는 앱입니다.' At the bottom, there are two buttons: '취소' and '저장'.

- 앱 이름 : 임의 설정
- 사업자명 : 임의 설정 (팀명 입력하였음)
- ☒ 서비스 이용이 제한되는 카테고리, 금지된 내용, 금지된 행동 관련 운영 정책을 위반하지 않는 앱입니다. 체크

(3) 전체 애플리케이션 목록 > 앱 설정 > 요약정보

REST API 를 spring boot app의 src > main > resources > application.yml 의 **client-id** 에 입력 (application.yml 참고 요망)

```
spring:
  security:
```

앱 키

네이티브 앱 키

REST API 키

JavaScript 키

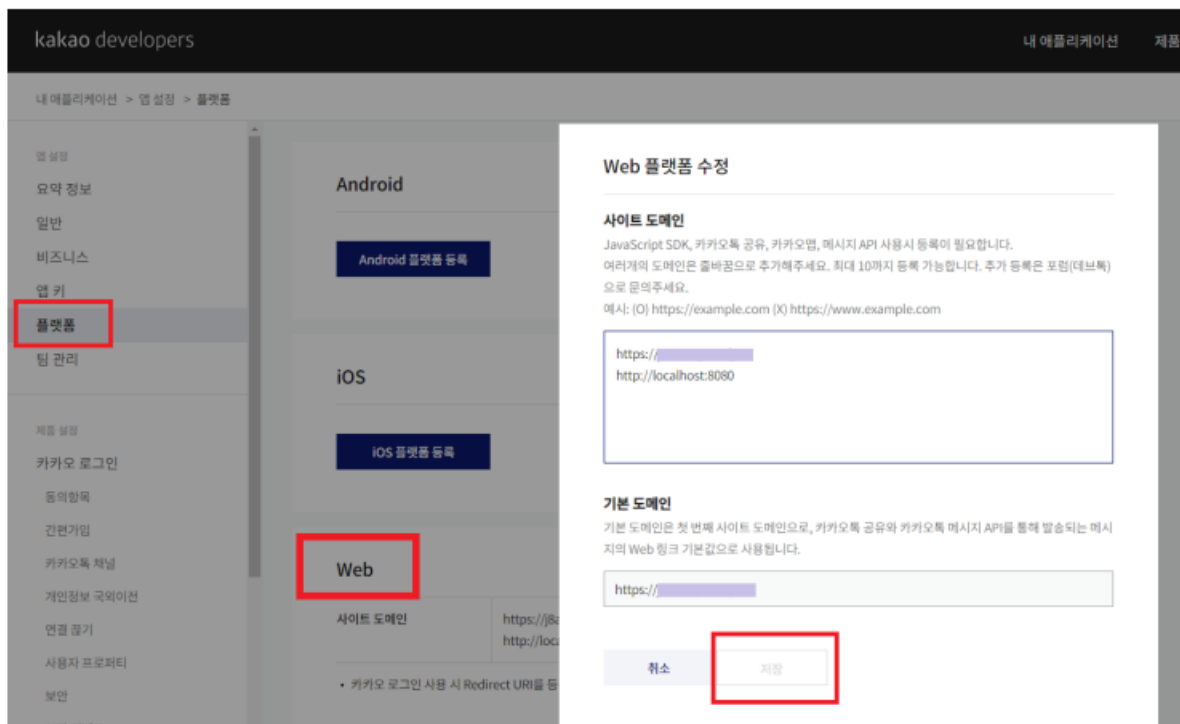
Admin 키

```

oauth2:
  client:
    registration:
      kakao:
        client-id:

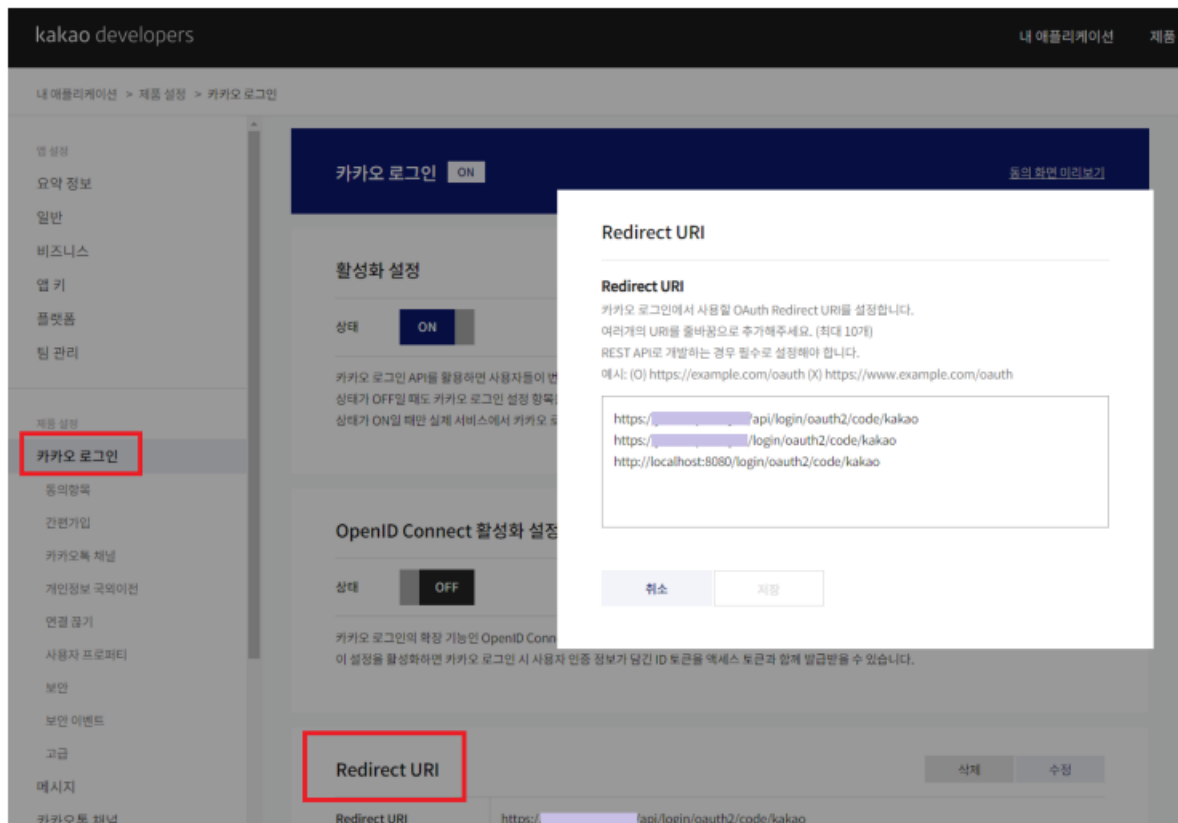
```

(4) 내 애플리케이션 > 앱 설정 > 플랫폼 > Web



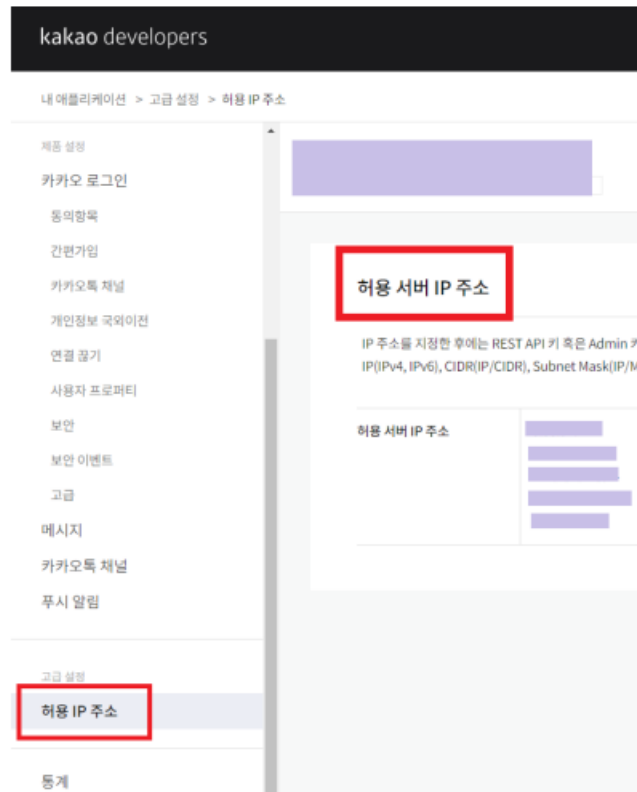
- 사이트 도메인 등록 : 서비스 도메인, 로컬 등등 kakao OAuth를 사용할 도메인 등록

(5) 내 애플리케이션 > 제품 설정 > 카카오 로그인



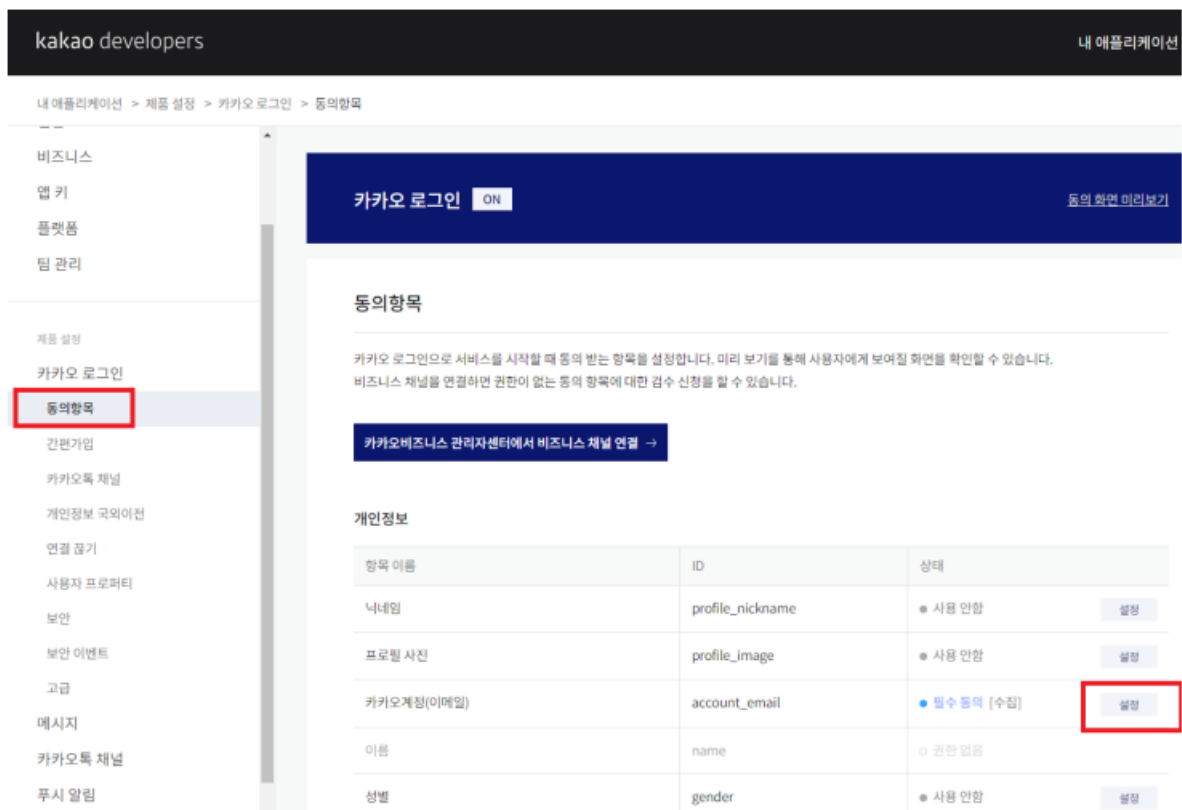
- Redirect URI 등록
- `https://[서비스도메인]/login/oauth2/code/kakao` : 해당 uri가 Spring Security의 기본 format

(6) 내 애플리케이션 > 고급 설정 > 허용 IP 주소



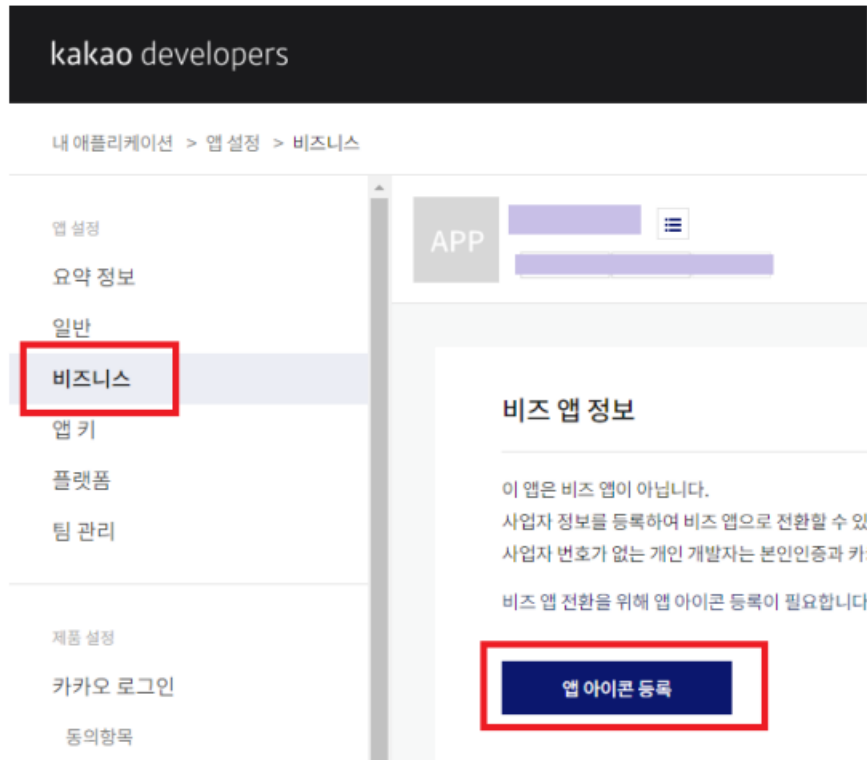
- 허용 서버 IP 주소 등록

(7) 내 애플리케이션 > 제품 설정 > 카카오 로그인 > 동의항목

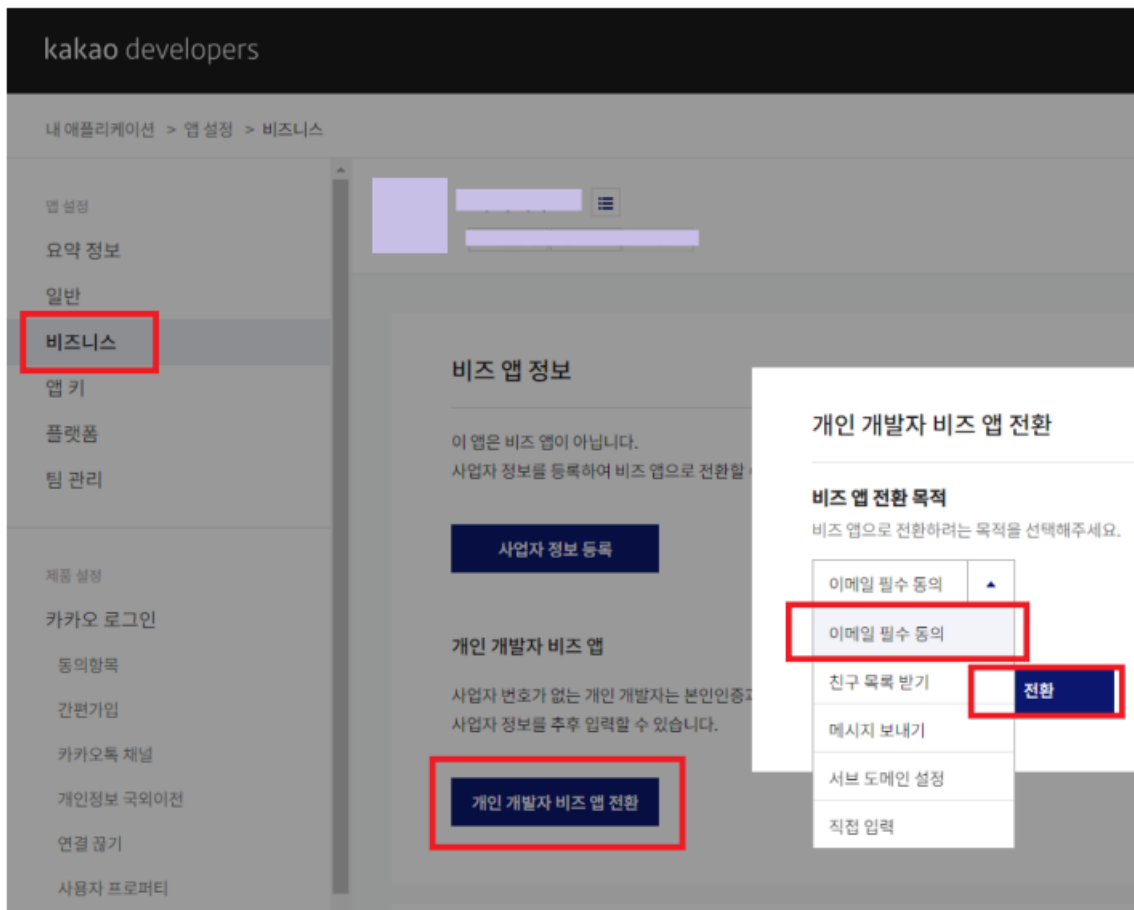


- 필요한 유저 정보에 대해 동의 항목 설정 추가
- 카카오회정(이메일) 추가 시 비즈앱 등록 필수

(7-1) 내 애플리케이션 > 앱 설정 > 비즈니스



- 비즈 앱 등록 시 앱 아이콘 등록 필수



- 개인 개발자 비즈니스 앱 전환 > 비즈니스 앱 전환 목적 선택