



# 1. 개발 환경

## 형상 관리

- Gitlab

## 이슈 관리

- Jira

## CI/CD

- jenkins(2.375.2)
- Docker

## IDE

- IntelliJ 2022 1.3
- Android Studio

## Server

- AWS EC2
  - Ubuntu 20.04LTS
  - Docker 20.10.18

## Communication

- Notion
- Discord
- Google Meet

## UI/UX

- Figma

## OS

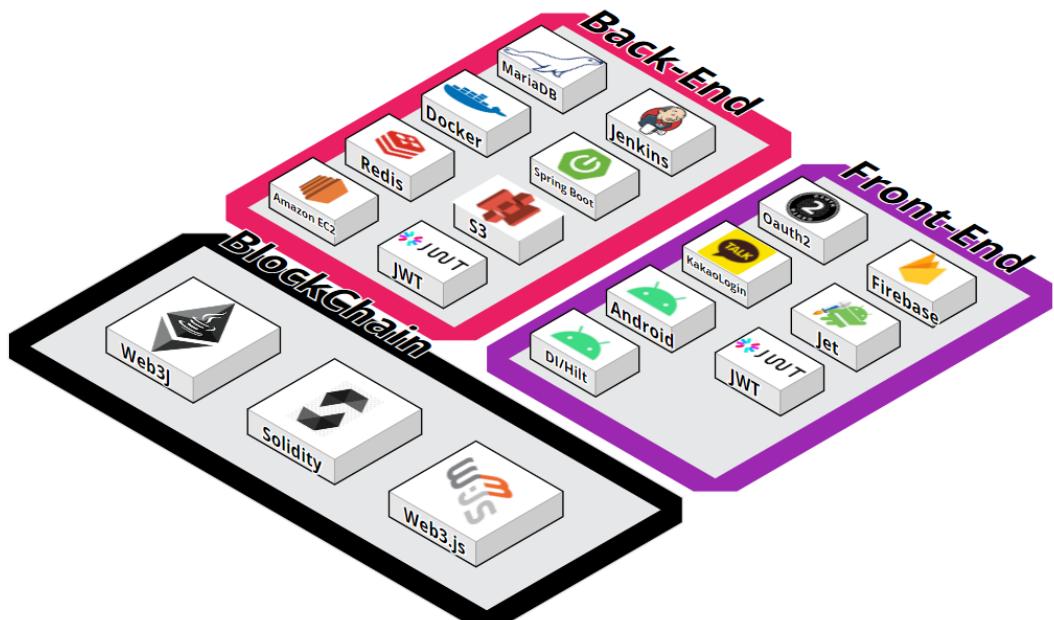
- Window 10
- MacOS Monterey

## DataBase

- Ver 15.1 Distrib 10.10.2-MariaDB
- MySQL WorkBench 6.X
- Redis

## 기타 편의 툴

- PostMan 9.28.1



## Front-End

- LiveData
- ViewModel
- DataBinding
- ViewPagerAdapter
- OkHttp3
- Retrofit2
- Coroutine
- Navigation
- Glide
- Hilt
- Version catalog
- Android Keystore
- bootpay
- exoplayer

## Back-End

- Java Open-JDK 11.0.15
- Gradle 8.0
- Spring Boot 2.7.9
  - Spring Data JPA
  - Lombok
  - Swagger 3.0.0
- Spring Security 2.7.9
- JUnit5 5.7.0
- Docker 4.15.0
- Jenkins:lts
- S3
- Boot pay 1.0.13
- H2 Database 2.7.9
- Redis 7.0.8
- MariaDB 10.10.2

## Blockchain

- Solcjs (0.8.19)
- Web3j (4.9.7)
- Remix IDE
- MetaMask
- Solidity 0.8.0



## 2. 배포서버 환경 구성(CI/CD)

### 목차

- [방화벽 설치](#)
- [Docker 및 docker-compose 설치](#)
- [도커 실행](#)
- [jenkins 설치 및 실행](#)
- [jenkins 초기 설정](#)
- [Jenkins 플러그인 추가 설치](#)
- [jenkins Credential등록](#)
- [jenkins 세부 설정](#)
- [Gitlab WebHook설정](#)
- [Jenkins 설정 후 서버 빌드 및 배포 방법](#)

### ▼ 방화벽 설치

#### 1. 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

1. **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.
2. **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.
3. **curl** : 이 도구는 명령행에서 데이터 전송을 수행할 수 있도록 합니다. 여러 프로토콜을 지원하며, URL 문법을 사용하여 데이터를 전송하거나 받을 수 있습니다.
4. **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.
5. **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.

#### 4. Docker의 GPC key 인증

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

## 5. Docker Repository 등록

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

## 5. 도커 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/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 옮겨 놓는 것이다.
# 결과적으로 "/usr/local/bin/" 경로에 "docker-compose"라는 이름의 파일로 다운된다.
# 참고) https://github.com/docker/compose/releases에서 최신 버전 확인이 가능하다.
# 파일별 저장 url을 확인 후 작성하셔야 합니다.

sudo chmod +x /usr/local/bin/docker-compose # chmod 를 통해서 실행이 가능하게 세팅

docker-compose -v # docker-compose 명령이 제대로 먹히는지 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

### a. local gradle 버전 확인

```
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 컨테이너로 올리기

## 13. 원격 데이터베이스 생성하기

```
ubuntu@ip-172-26-3-38:~/S08P22D208/Server$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
45b1db8c07d1        server_module-batch   "java -Duser.timezone..."   3 minutes ago      up 3 minutes
5034bf51ff93        mariadb            "docker-entrypoint.s..."   3 minutes ago      up 3 minutes ago
51570b204727        redis:alpine        "docker-entrypoint.s..."   3 minutes ago      up 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
```

## ▼ 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 옮겨 놓는 것이다.  
# 결과적으로 "/usr/local/bin/" 경로에 "docker-compose"라는 이름의 파일로 다운된다.  
# 참고) https://github.com/docker/compose/releases에서 최신 버전 확인이 가능하다.  
# 최신 버전을 설치하고 싶다면 위 명령어에 보이는 1.28.5라는 버전 숫자를 바꿔주면 된다!  
  
sudo chmod +x /usr/local/bin/docker-compose # chmod를 통해서 실행이 가능하게 세팅  
  
docker-compose -v # docker-compose 명령이 제대로 먹히는지 확인한다.
```

## ▼ 도커 실행

```
sudo systemctl enable docker
```

```
sudo service docker start
```

## ▼ 젠킨스 설치 및 실행

1. Dockerfile을 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-archive-keyring.gpg
    && echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/debian $"
    && 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 .
```

2. 이미지 확인

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

2. volume 만들기

```
docker volume create jenkins
```

3. 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
```

## ▼ 젠킨스 초기 설정

1. <http://서버아이피:8080>으로 접속 한다.
2. 젠킨스에 처음 접속하면 초기 관리자 계정의 비밀번호를 입력하라고 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

## Getting Started

# CUSTOMIZE JENKINS

Plugins extend Jenkins with additional features to support many different needs.

**Install suggested plugins**

Install plugins the Jenkins community finds most useful.

**Select plugins to install**

Select and install plugins most suitable for your needs.

Jenkins 2.289

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

Getting Started				
<h1>Getting Started</h1> 				
Folders	OWASP Markup Formatter	Build Timeout	Credentials Binding	Workspace Cleanup
✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	Ant ** JavaScript GUI Lib: ACE Editor bundle ** JavaScript GUI Lib: jQuery bundles (jQuery and jQuery UI) ** Pipeline: SCM Step ** Pipeline: Groovy ** Pipeline: Job ** Apache HttpComponents Client 4.x API ** Display URL API Mailer ** Pipeline: Basic Steps Gradle ** Pipeline: Milestone Step ** Jackson 2 API ** Pipeline: Input Step ** Pipeline: Stage Step ** Pipeline: Graph Analysis ** Pipeline: REST API ** JavaScript GUI Lib: Handlebars bundle ** JavaScript GUI Lib: Moment.js bundle Pipeline: Stage View ** Pipeline: Build Step ** Pipeline: Model API ** - required dependency
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle	
⌚ Pipeline	⌚ GitHub Branch Source	⌚ Pipeline: GitHub Groovy Libraries	⌚ Pipeline: Stage View	
⌚ Git	⌚ Subversion	⌚ SSH Slaves	⌚ Matrix Authorization Strategy	
⌚ PAM Authentication	⌚ LDAP	⌚ Email Extension	✓ Mailer	

#### ▼ Jenkins 플러그인 추가 설치

The screenshot shows the Jenkins Plugin Manager page. The top navigation bar includes links for Dashboard, Jenkins 관리, and Plugin Manager. On the left, there are four tabs: Updates, Available plugins (which is selected and highlighted in grey), Installed plugins, and Advanced settings. A search bar at the top right contains the placeholder text 'Search available plugins'. Below the search bar is a table with columns for 'Install', 'Name' (sorted by descending order as indicated by the downward arrow), and 'Released'. The table lists several available plugins, such as 'Ansible', 'Apache Ant', 'Apache Maven', 'Apache Velocity', 'Antlr', 'Apache Geronimo JAX-WS', 'Apache Geronimo JAX-RPC', 'Apache Geronimo JAX-RS', 'Apache Geronimo JAX-B', 'Apache Geronimo JAX-WS 2.3', 'Apache Geronimo JAX-RPC 2.3', 'Apache Geronimo JAX-RS 2.3', 'Apache Geronimo JAX-B 2.3', 'Apache Geronimo JAX-WS 2.2', 'Apache Geronimo JAX-RPC 2.2', 'Apache Geronimo JAX-RS 2.2', 'Apache Geronimo JAX-B 2.2', 'Apache Geronimo JAX-WS 2.1', 'Apache Geronimo JAX-RPC 2.1', 'Apache Geronimo JAX-RS 2.1', 'Apache Geronimo JAX-B 2.1', 'Apache Geronimo JAX-WS 2.0', 'Apache Geronimo JAX-RPC 2.0', 'Apache Geronimo JAX-RS 2.0', 'Apache Geronimo JAX-B 2.0', 'Apache Geronimo JAX-WS 1.1', 'Apache Geronimo JAX-RPC 1.1', 'Apache Geronimo JAX-RS 1.1', 'Apache Geronimo JAX-B 1.1', 'Apache Geronimo JAX-WS 1.0', 'Apache Geronimo JAX-RPC 1.0', 'Apache Geronimo JAX-RS 1.0', and 'Apache Geronimo JAX-B 1.0'. Each plugin entry includes a 'Install' button.

## 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 명령어를 사용해 재실행해준다.

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

#### ▼ 젠킨스 Credential등록

- ### 1. Jenkins 관리 → Manage Credentials에 들어간다

## Security

The screenshot shows the Jenkins Security page. It includes links for 'Configure Global Security', 'Manage Users', and 'Configure Credential Providers'. The 'Manage Credentials' link is highlighted with a red box.

- Configure Global Security
- Manage Credentials
- Configure Credential Providers

Manage Users

Configure credentials

Configure the credential providers and types

## 2. System 선택

# Credentials

The screenshot shows the Jenkins Credentials page. The 'Domain' dropdown is set to 'System' (highlighted in yellow), and the 'Store' dropdown is set to 'T' (Username). The table below lists a single credential entry.

T	P	Store ↓	Domain
		System	(global)

## 3. 우측 상단에 Add Credentials 클릭

The screenshot shows the Jenkins Global credentials (unrestricted) page. A single credential named 'uberamen5ch1308' is listed. The 'Add Credentials' button is located in the top right corner.

ID	Name	Kind	Description
	uberamen5ch1308	uberamen5ch1308/***** (gitlab)	Username with password

## 4. 우선 Gitlab 프로젝트 Repository에 들어가서 accesstoken을 발급받는다

The screenshot shows the GitLab Access Tokens creation page. The left sidebar shows project navigation. The main form allows creating a project access token with fields for Token name, Expiration date (set to 2023-03-18), Select a role (Guest), and Select scopes (api, read\_api, read\_repository). A note states that scopes set the permission levels granted to the token.

Project Access Tokens

Add a project access token

Enter the name of your application, and we'll return a unique project access token.

Token name

Expiration date

2023-03-18

Select a role

Guest

Select scopes

Scopes set the permission levels granted to the token. [Learn more](#).

api  
Grants complete read and write access to the scoped project API, including the Package Registry.

read\_api  
Grants read access to the scoped project API, including the Package Registry.

read\_repository

## 5. 발급 받은 accessToken을 Credential을 만들때 password로 입력해주고 나머지 부가 정보들도 입력해주고 Create해준다

New credentials

Kind: Username with password

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

Username: übermen5ch1308 Gitlab username

Treat username as secret:

Password: project accessstoken 입력

ID: Gitlab ID 입력

Description:

**Create**

## ▼ 젠킨스 세부 설정

1. 젠킨스 로그인을 완료하고 새로운 Item추가를 클릭한다.
2. item이름을 입력하고 Freestyle project를 선택한다.

Enter an item name

campinity-develop<sup>2</sup>

» 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**

3. 소스 코드 관리 목록중 Git을 선택하고 Repository URL에 Gitlab프로젝트 URL을 입력하고 이전에 설정해둔 Credential로 선택해준다.

### 소스 코드 관리

None

Git [?](#)

#### Repositories [?](#)

Repository URL [?](#)

<https://lab.ssafy.com/s08-webmobile4-sub2/S08P12D101.git> (깃랩 repository URL)

! Please enter Git repository.

#### Credentials [?](#)

- none - GitLab repository token을 기반으로 만든 Credential등록

[+ Add](#)

[고급...](#)

4. 바로 아래에 어떤 브랜치에서 commit들고와서 Integration 시킬지 branch를 명시해준다. (deploy-be로 설정)

Branches to build ?

Branch Specifier (blank for 'any') ?

X

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

The screenshot shows the 'Build Steps' configuration section in Jenkins. It contains two main sections:

- Invoke Gradle script**:
  - Gradle Version**: gradle 8.0
  - Tasks**: build -p /var/jenkins\_home/workspace/campinity-develop/Server -x test
  - Buttons**: 고급... (Advanced) and Apply
- Execute shell**:
  - Command**:

```
cd Server
docker-compose up --build -d
```
  - See the list of available environment variables**
  - Buttons**: 저장 (Save) and Apply

## ▼ Gitlab WebHook설정

1. Gitlab WebHook 탭에 들어가서 설정값들을 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.

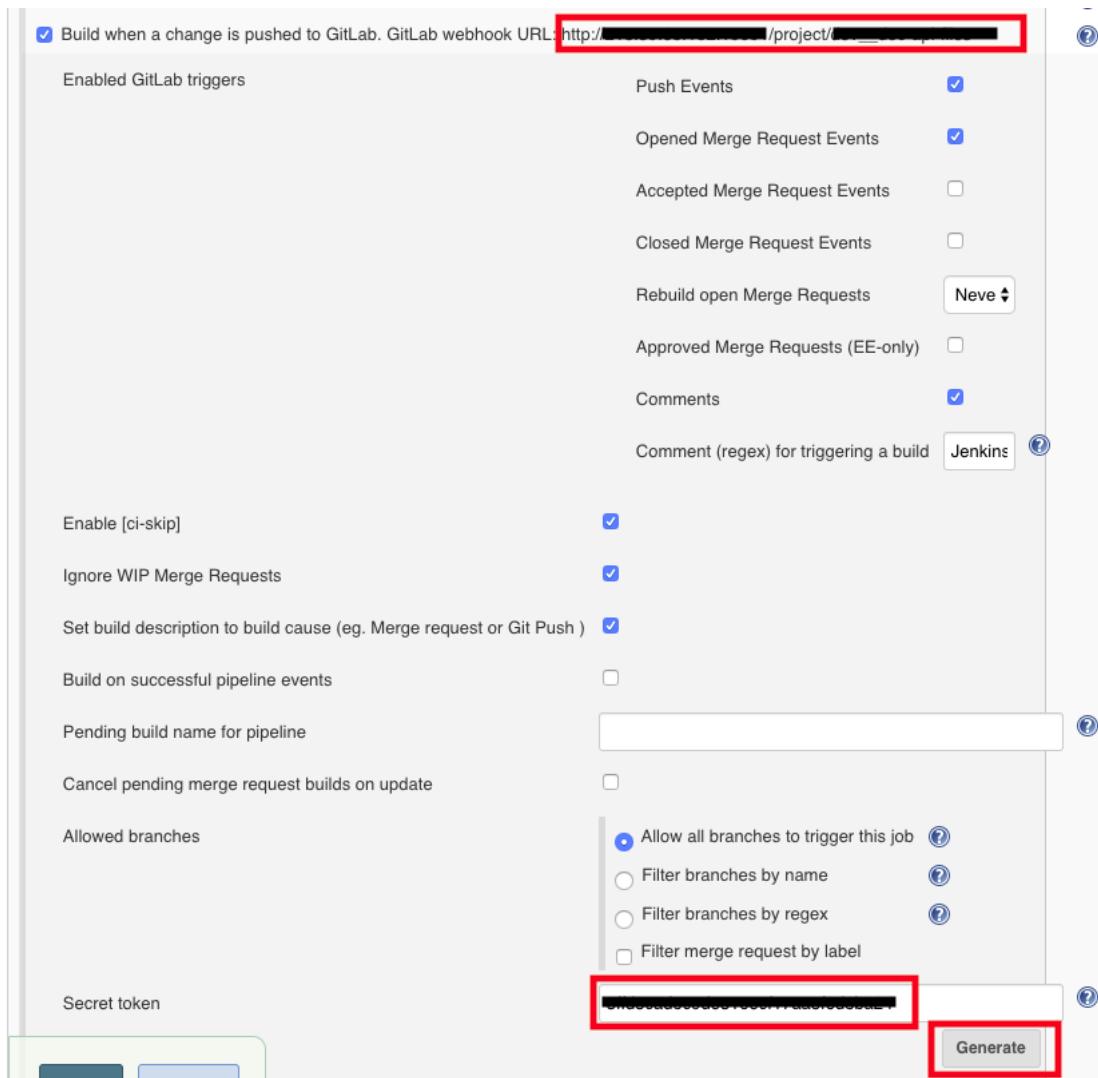
Confidential comments  
A comment is added to a confidential issue.

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. 설정값중 Secret token은 jenkins서버에 접속해서 item(campinity-develop)→구성→빌드유발→고급→Secret token→generate 클릭해서 토큰을 발급받고 입력해준다.



### 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
<a href="#">View details</a>			

## ▼ Jenkins 설정 후 서버 빌드 및 배포 방법

### 자동으로 빌드 및 배포

Jenkins에 등록한 project URL에 Merge Request가 accept되면 자동으로 빌드 및 배포가 된다.

### 수동으로 빌드 및 배포

The screenshot shows the Jenkins dashboard for the 'campinity-develop' project. At the top, there's a breadcrumb navigation: Dashboard > campinity-develop >. Below it, a horizontal menu bar contains several items: 상태 (Status), 변경 사항 (Changes), 작업공간 (Workspaces), **지금 빌드** (Build Now) which is highlighted in yellow, 구성 (Configure), Project 삭제 (Delete Project), and Rename. Underneath this is a 'Build History' section with a table showing the last six builds. The table includes columns for build number, status, duration, and timestamp.

#	상태	걸린 시간	마지막 업데이트
225	Success	25 min	2023-10-16 10:45:23
225	Success	25 min	2023-10-16 10:45:23
225	Success	25 min	2023-10-16 10:45:23
131	Failure	5 days 15 hr	2023-10-16 09:45:23
155	Failure	3 days 3 hr	2023-10-16 09:45:23
225	Success	25 min	2023-10-16 10:45:23

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

## 참고 사항

위의 과정을 그대로 따라서 거쳐온다면, 서버가 제대로 작동하지 않을것이다. 그 이유는 MariaDB 컨테이너에 Database가 생성되지 않은 상태이기 때문이다.

따라서 EC2에 원격접속 후 DB관련 환경설정을 해주어야한다.

[3. MariaDB 환경설정](#)



## 3. MariaDB 환경설정



MariaDB Container에 접속 후 접속할 유저 생성

### 1. EC2 접속

```
ssh -i I8D101.pem ubuntu@i8d101.p.ssafy.io
```

### 2. MariaDB container 접속

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

### 3. MariaDB 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

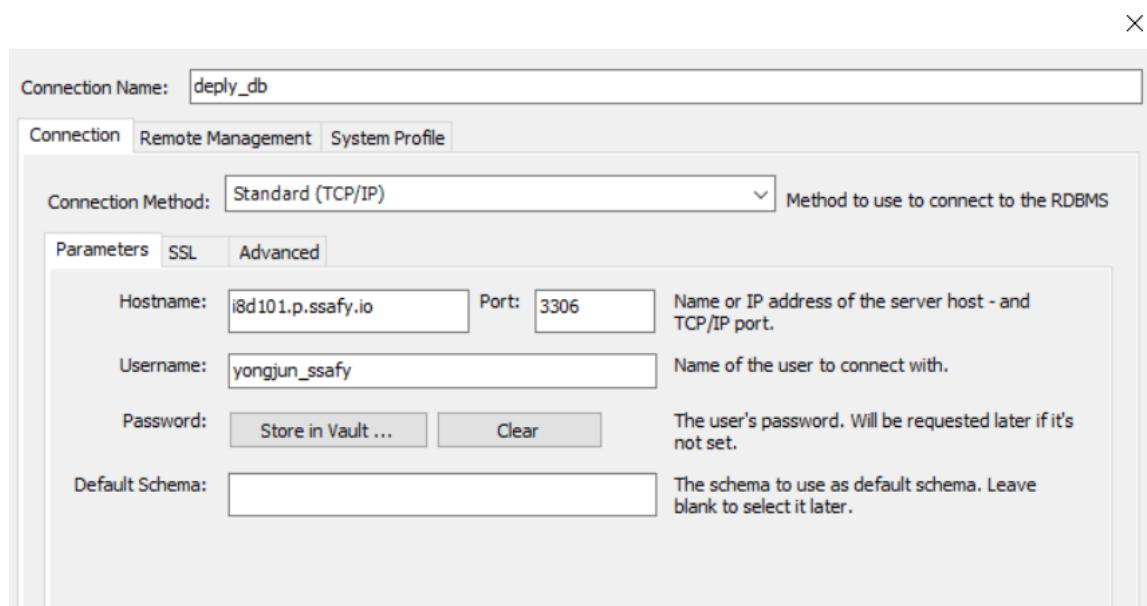
```
flush privileges;
```



## MariaDB 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



### 2. 접속 후 project repository/exec 폴더에있는 backup.sql을 다운받아서 sql을 execute 해준다.



## 4. 빌드 및 테스트 방법

### 블록체인(Local)

[Truffle 환경 구축](#)

[Truffle을 활용한 컨트랙트 배포](#)

[트랜잭션 테스트 방법](#)

### 블록체인(SSAFY-Testnet)

[Remix IDE를 활용한 배포 및 테스트 방법](#)

### 백엔드

[1. Repository clone](#)

[2. core application.yml 수정](#)

[4. Gradle을 이용해 빌드\(jar파일 생성\)](#)

[5. docker-compose로 container띄우기](#)

[6. docker-compose로 container띄우기](#)

### 안드로이드

[1. Repository clone](#)

[2. app module: build.gradle](#)

[4. libs.versions.toml](#)

[5. import 구문을 통해 web3j 라이브러리 활용 확인](#)

## 블록체인(Local)

### ▼ Truffle 환경 구축



Python 3.x, node.js lts 버전 설치 필요

1. Truffle 설치 (**npm install -g truffle@5.0.2**)

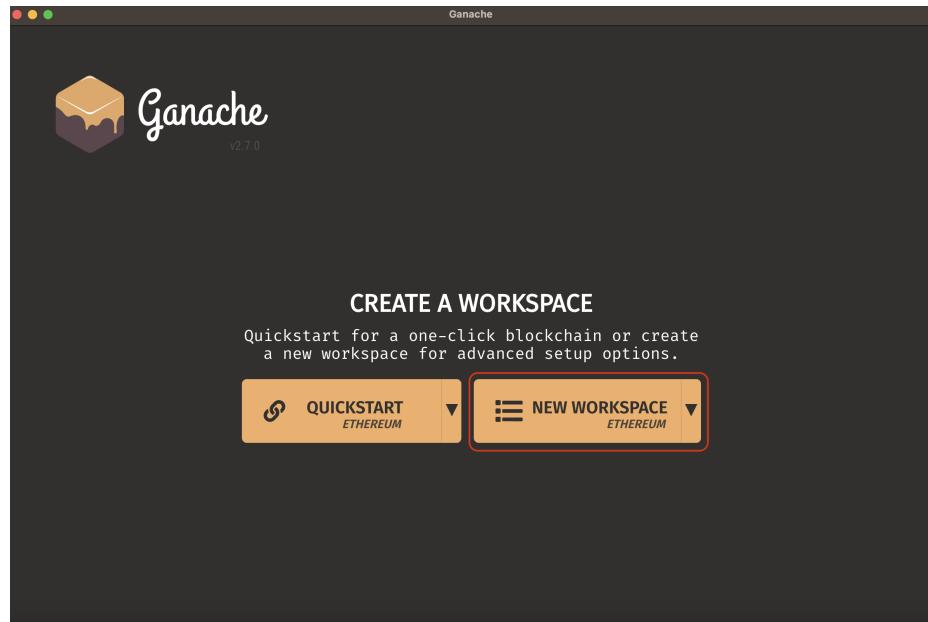
2. 임의 폴더 생성 (ex)C://truffle/kkaddak)

3. truffle init (아래와 같이 contracts, migrations, test, package.json, truffle-config 생성됨)

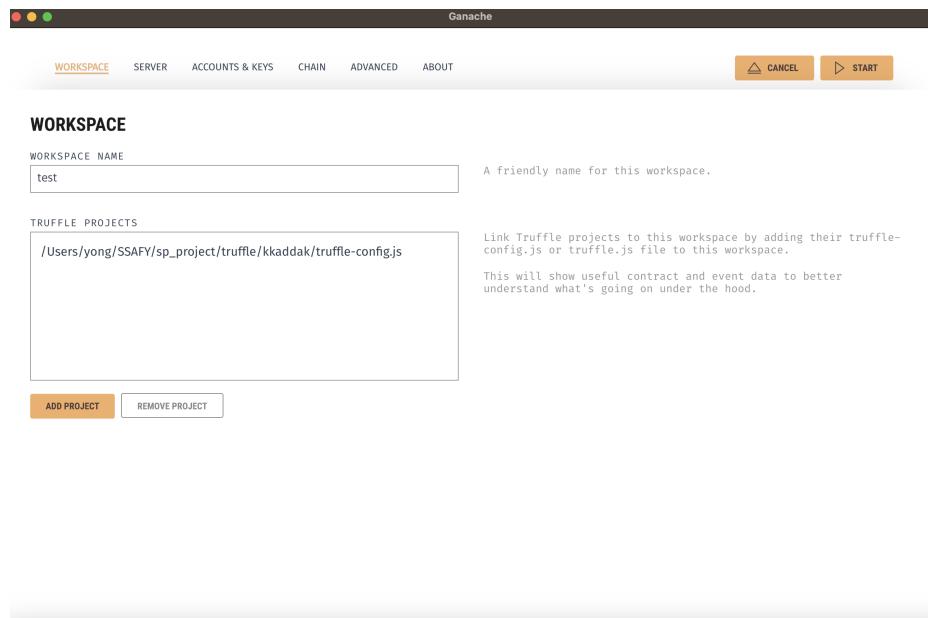
이름	수정일	크기	종류
> build	오늘 오후 8:34	--	폴더
> contracts	오늘 오후 8:30	--	폴더
> migrations	오늘 오후 8:40	--	폴더
package-lock.json	오늘 오후 8:27	201바이트	JSON file
package.json	오늘 오후 8:27	255바이트	JSON file
> test	오늘 오후 8:26	--	폴더
truffle-config.js	오늘 오후 8:28	6KB	JavaScr...t source

4. npm init 명령어 실행(정상적으로 완료됐다면 package.json파일 생성됨)

5. npm install
6. Ganache 설치 (<https://www.trufflesuite.com/ganache>)
7. Ganache 실행 후 workspace 생성



8. 이전에 만든 폴더에 존재하는 truffle-config.js를 import시켜줌



9. 정상적으로 됐다면 아래와 같은 화면이 보임

The screenshot shows the Ganache interface with the following account details:

ADDRESS	BALANCE	TX COUNT	INDEX	
0x1e688712c90DDb3ed8FbC26C8a6c44359dF4DdE8	100.00 ETH	0	0	
0xfd7DD4Fb18e92d30018F5DB429839Ee8cDD7F2ea	100.00 ETH	0	1	
0x9Ff371c1023783F50eF19ECe32B554721EdE10D9	100.00 ETH	0	2	
0x86EF2d3FD57076d073038B94925111210eb6292b	100.00 ETH	0	3	
0xa51608eCa2cac161877B1FaCa41CE1c600eDdD1F	100.00 ETH	0	4	
0x746667E584f74A388993cFB1D6d6754a3DE125e1	100.00 ETH	0	5	
0x6915F14FB8cD4A0E64Be944f0543082D46a02c8a	100.00 ETH	0	6	

## ▼ Truffle을 활용한 컨트랙트 배포

- Truffle init을 실행한 폴더하위에 contracts폴더에 .sol파일을 작성

The screenshot shows the VS Code editor with the following content:

```

KATTToken.sol — kkaddak
JS truffle-config.js    KATTToken.sol    JS 2_deploy_contracts.js
contracts > KATTToken.sol
pragma solidity ^0.8.19;

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount, string calldata transferType) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event MsgSender(address msgSender);
}

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL
1: zsh
Summary
=====
> Total deployments: 1
> Final cost: 0.0055656585 ETH
~ /SSAFY/sp_project/truffle/kkaddak/

```

- 터미널에서 truffle compile 명령어 실행

- /build/contracts 폴더안에 파일명.json 파일이 생성되어있는것을 확인

The screenshot shows the VS Code interface with the following details:

- EXPLORER** view: Shows the project structure under "KKADDAK". A red box highlights the "build/contracts" folder, which contains three JSON files: IERC20.json, KATTOKEN.json, and SafeMath.json.
- RIGHT PANEL**: An open file is "KATTOKEN.sol" (Solidity code). The code defines an interface for IERC20 and a contract for KATTOKEN. It includes functions like totalSupply, balanceOf, transfer, allowance, approve, and transferFrom, along with an event Transfer.

#### 4. migrations폴더로 가서 2\_deploy\_contracts.js파일 생성

```
// 2_deploy_contracts.js
const KATTOKEN = artifacts.require("KATTOKEN");

module.exports = function(deployer) {
  deployer.deploy(KATTOKEN, 'KATTOKEN', 'KAT', 8); // KATTOKEN 컨트랙트 생성자 파라미터를 적절히 넣어준다
};
```

#### 5. 터미널창에서 truffle migrate 명령어 실행

```
Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Starting migrations...
=====
> Network name:    'development'
> Network id:      5777
> Block gas limit: 6721975 (0x6691b7)

2_deploy_contracts.js
=====

Deploying 'KATTOKEN'
-----
> transaction hash: 0x1de10bd16bb27e848f8ec00a44fd1f3760946960a952057e9228ed776ab5af9c
> Blocks: 0          Seconds: 0
> contract address: 0x7b3EE73A1d615A9C3F2d1F96E20df96234515353
> block number:     1
> block timestamp:  1680695134
> account:          0xbb381F50B90fdc5DE341de84A14d6cB1dE9c3b9c
> balance:           99.9944343415
> gas used:         1649084 (0x1929bc)
> gas price:        3.375 gwei
> value sent:       0 ETH
> total cost:       0.0055656585 ETH

> Saving artifacts
-----
> Total cost:       0.0055656585 ETH
```

```
Summary
=====
> Total deployments:    1
> Final cost:          0.0055656585 ETH
```

위와 같이 배포된것을 확인할 수 있고 다양한 정보를 확인가능하다

## ▼ 트랜잭션 테스트 방법

1. Ganache를 실행하고 계정의 주소와 private key를 확인한다

- ## 2. 배포된 컨트랙트 주소를 확인한다

The screenshot shows the Ganache interface with the following details:

- Accounts:** CURRENT BLOCK 1, GAS PRICE 20000000000, GAS LIMIT 6721975, HARDFORK MERGE, NETWORK ID 5777, RPC SERVER HTTP://127.0.0.1:7545, MINING STATUS AUTOMINING.
- Logs:** WORKSPACE KADDAK.
- Switch:** SWITCH.
- Contracts:** kkaddak, IERC20, KATTOKEN, SafeMath.

The **kkaddak** contract is currently selected, showing its deployment path: /Users/yong/SSAFY/sp\_project/truffle/kkaddak.

NAME	ADDRESS	TX COUNT
IERC20	Not Deployed	0
KATTOKEN	0x7b3EE73A1d615A9C3F2d1F96E20df96234515353	0
SafeMath	Not Deployed	0

3. 테스트 코드가 참조하는 yml파일을 적절하게 수정한다

~/S08P22D208/Server/module-api/src/test/resources/application.yml

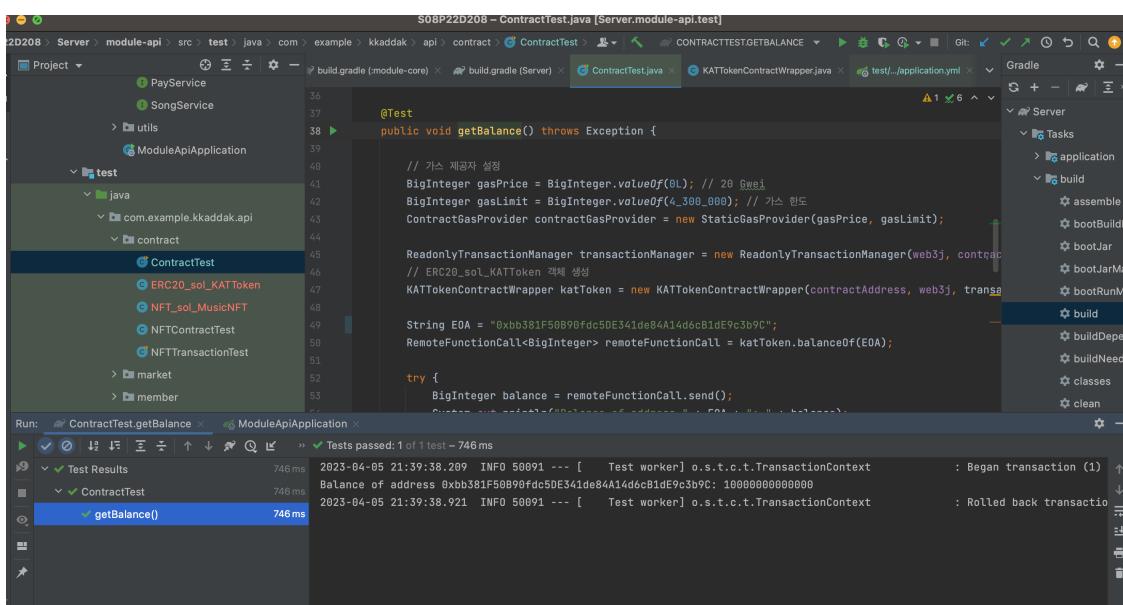
- #### 4. 빌드 및 테스트 방법

```

ethereum:
contract:
  wallet-address: '0x7b3EE73A1d615A9C3F2d1F96E20df96234515353'
  song-address: ''
  nft-address: '0x0C0391FF59A532a51cf8E10F3C0632401EA0b4B8'
  rpc-url: 'http://127.0.0.1:7545'
  admin-address: '0xe5ece0c6abecb6f0328734651c337ab3a524ddb'
  private-key: '0xe613f878ed3ff9b8a7e1f284cad62e594ea3b1cb324fe692df0bf80ee371316e'
  exchange-rate: 1000000

```

#### 4. 테스트하고자 하는 컨트랙트의 테스트코드를 실행해본다 (잔액조회 메소드 성공적으로 실행완료)



## 블록체인(SSAFY-Testnet)

### ▼ Remix IDE를 활용한 배포 및 테스트 방법

1. Remix IDE 실행
2. solidity 코드를 작성한다 (contracts폴더에)

```

// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract MusicCopyright {
    struct Song {
        string title;
        string artist;
        string cover;
        string songFile;
        uint256 uploadTime;
        uint256 combination;
        address owner;
    }

    mapping(uint256 => Song) public songs;
    uint256 public songCount;

    event SongRegistered(uint256 songId, string title, string artist, string cover, string songFile, uint256 uploadTime);

    function registerSong(string memory _title, string memory _artist, string memory _cover, string memory _songFile, uint256 _uploadTime) public {
        songCount++;
        songs[songCount] = Song(_title, _artist, _cover, _songFile, _uploadTime, _combination);
        emit SongRegistered(songCount, _title, _artist, _cover, _songFile, _uploadTime, _combination);
        return songCount;
    }
}

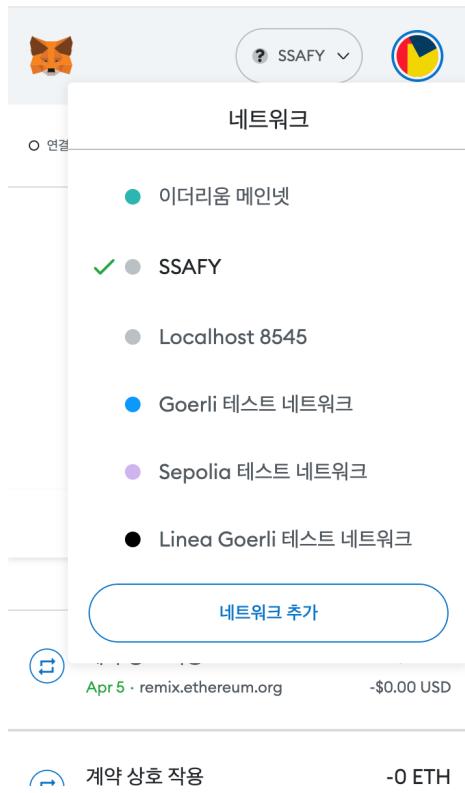
```

3. 왼쪽 메뉴탭에서 세번째를 클릭하고 컴파일한다

4. 싸피 테스트 네트워크에 연결한다 (MetaMask 필요)

5. 크롬 확장 프로그램으로 설치된 MetaMask를 실행하여 네트워크 추가 버튼 클릭

4. 빌드 및 테스트 방법



6. 아래와 같은 화면이 나오게되고 네트워크 수동 추가 클릭

#### 네트워크 > 네트워크 추가

주요 네트워크 목록에서 추가하거나 네트워크를 직접 추가합니다. 신뢰할 수 있는 엔티티만 이용하세요.

인기 사용자 정의 네트워크

	Arbitrum One	<a href="#">추가</a>
	Aurora Mainnet	<a href="#">추가</a>
	Avalanche Network C-Chain	<a href="#">추가</a>
	BNB Smart Chain (previously Binance Smart Chain Mainnet)	<a href="#">추가</a>
	Celo Mainnet	<a href="#">추가</a>
	Fantom Opera	<a href="#">추가</a>
	Harmony Mainnet Shard 0	<a href="#">추가</a>
	Optimism	<a href="#">추가</a>
	Palm	<a href="#">추가</a>
	Polygon Mainnet	<a href="#">추가</a>

#### 네트워크 수동 추가

7. SSAFY 네트워크 정보를 입력하고 저장 클릭

네트워크 > 네트워크 추가 > 네트워크 수동 추가

**i** 악성 네트워크 공급업체는 블록체인 상태를 거짓으로 보고하고 네트워크 활동을 기록할 수 있습니다. 신뢰하는 맞춤형 네트워크만 추가하세요.

네트워크 이름

SSAFY

새 RPC URL

<https://rpc.ssafy-blockchain.com>

이 URL은 현재 SSAFY 네트워크에서 사용됩니다.

체인 ID **i**

31221

통화 기호

ETH

현재 티커 기호 확인 데이터를 사용할 수 없습니다. 입력한 기호가 올바른지 확인하세요. 이는 네트워크에 표시되는 전환율에 영향을 미칩니다.

블록 탐색기 URL (옵션)

취소

저장

8. 네트워크가 SSAFY로 설정됐는지 확인하고 다시 Remix IDE로 이동

9. ENVIRONMENT 항목에서 Injected Provider - MetaMask 선택

DEPLOY & RUN TRANSACTIONS ✓

ENVIRONMENT ▾

- Injected Provider - MetaMask
- Injected Provider - MetaMask**
- Remix VM (Merge)
- Remix VM (London)
- Remix VM (Berlin)
- Remix VM - Mainnet fork
- Remix VM - Sepolia fork
- Remix VM - Goerli fork
- Remix VM - Custom fork
- Custom - External Http Provider
- Dev - Hardhat Provider
- Dev - Ganache Provider
- Dev - Foundry Provider
- L2 - Optimism Provider
- L2 - Arbitrum One Provider
- Wallet Connect

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract MusicCopyright {
    struct Song {
        string title;
        string artist;
        string cover;
        string songFile;
        uint256 uploadTime;
        uint256 combination;
        address owner;
    }

    mapping(uint256 => Song) public songs;
    uint256 public songCount;

    event SongRegistered(uint256 songId, string title, string artist, string cover, string songFile, uint256 uploadTime, uint256 combination, address owner);

    function registerSong(string memory _title, string memory _artist, string memory _cover, string memory _songFile, uint256 _uploadTime, uint256 _combination) public {
        songCount++;
        songs[songCount] = Song(_title, _artist, _cover, _songFile, _uploadTime, _combination);
        emit SongRegistered(songCount, _title, _artist, _cover, _songFile, _uploadTime, _combination);
        return songCount;
    }
}
```

## 10. 배포 버튼 클릭 후 확인

DEPLOY & RUN TRANSACTIONS ✓

ACCOUNT ▾

0x1E5...24DDB (0 ether)

GAS LIMIT

3000000

VALUE

0 Wei

CONTRACT (Compiled by Remix)

MusicCopyright - contracts/SongCopyri...

**Deploy** 1. 배포 버튼 클릭

Publish to IPFS

OR

At Address 0x0C0391FF59A532a51d8E...

Transactions recorded 15

Deployed Contracts

- IERC20 AT 0XF9...11A74 (BLOCKC...
- MUSICNFT AT 0X045...4BC86 (BLOCKC...
- MUSICCOPYRIGHT AT 0X4D9...3A2 (BLOCKC...
- MUSICNFT AT 0X0C0...0X4B8 (BLOCKC...

MetaMask Notification

https://remix.ethereum.org

개인 배포 \$0.00

세부 정보 데이터

예상 가스 요금 ⓘ \$0.00 0 ETH  
주전 사이트 최대 요금 0 ETH

합계 \$0.00 0 ETH  
금액 + 가스 요금 최대 금액: 0 ETH

2. 확인 클릭

기부 확인

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.0;

contract MusicCopyright {
    struct Song {
        string title;
        string artist;
        string cover;
        string songFile;
        uint256 uploadTime;
        uint256 combination;
        address owner;
    }

    mapping(uint256 => Song) public songs;
    uint256 public songCount;

    event SongRegistered(uint256 songId, string title, string artist, string cover, string songFile, uint256 uploadTime, uint256 combination, address owner);

    function registerSong(string memory _title, string memory _artist, string memory _cover, string memory _songFile, uint256 _uploadTime, uint256 _combination) public {
        songCount++;
        songs[songCount] = Song(_title, _artist, _cover, _songFile, _uploadTime, _combination);
        emit SongRegistered(songCount, _title, _artist, _cover, _songFile, _uploadTime, _combination);
        return songCount;
    }
}
```

listen on all transactions Search with transaction hash or address

to: MusicNFT.mintMusicNFT(address,string,string,string,string) 0x0C0...0b4B8 value: 0 wei data: 0x846...00000 logs: 1 hash: 0x877...c2bd3 creation of MusicCopyright pending...

## 11. 배포된 컨트랙트 확인

```

1 // SPDX-License-Identifier: GPL-3.0
2 pragma solidity ^0.8.0;
3
4 contract MusicCopyright {
5     struct Song {
6         string title;
7         string artist;
8         string cover;
9         string songFile;
10        uint256 uploadTime;
11        uint256 combination;
12        address owner;
13    }
14
15    mapping(uint256 => Song) public songs;
16    uint256 public songCount;
17
18    event SongRegistered(uint256 songId, string title, string artist, string co
19
20    function registerSong(string memory _title, string memory _artist, string m
21        songCount++;
22        songs[songCount] = Song(_title, _artist, _cover, _songFile, _uploadTime

```

Balance: 0 ETH

**registerSong** string \_title, string \_artist, string \_cover, string \_songFile, uint256 \_uploadTime

**songCount**

**songs** uint256

Low level interactions

CALldata

Transact

12. 배포된 컨트랙트의 함수 보이게되고 적절한 파라미터를 입력하면 트랜잭션 전송이 가능하다.

## 백엔드



### 환경 구성

- 운영체제 : Window 10
- 준비물 : IntelliJ, Docker Desktop, Git



### 빌드 및 테스트 방법

#### 1. Repository clone

```
git clone --single-branch --branch develop-be https://lab.ssafy.com/s08-blockchain-contract-sub2/S08P22D208.git
```

#### 2. core application.yml 수정

```
~/S08P22D208/Server/module-core/src/main/resources/application.yml
```

위 경로에 있는 파일을 열어서 `spring:profiles:active: local`로 수정한다

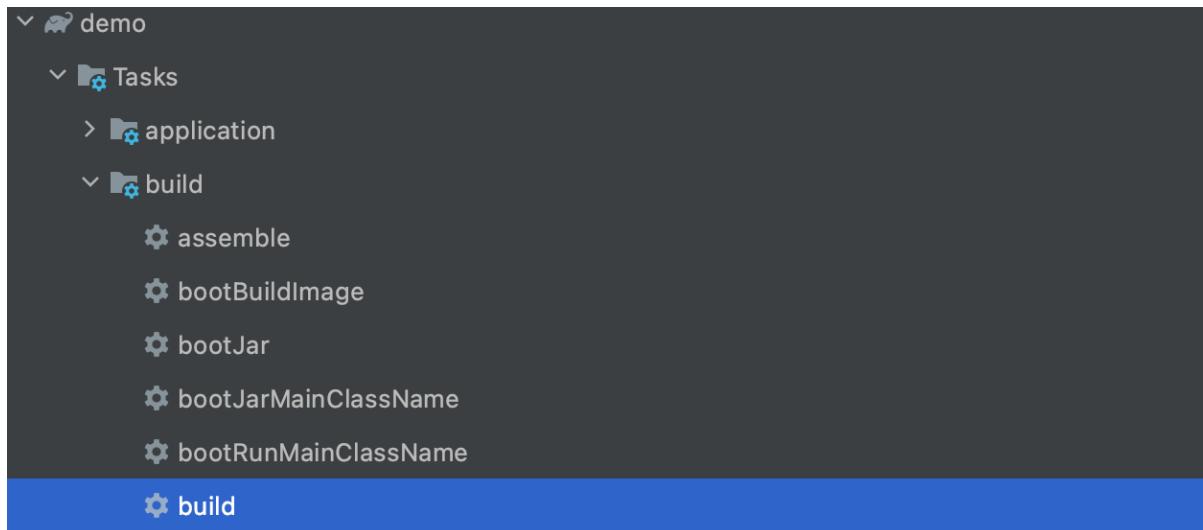
```

1   server:
2     port: 8087
3
4   spring:
5     # 빈 정의를 덮어쓰도록 허용하는 방법
6     # 임시 대책이기 때문에 근본대책 필요
7     profiles:
8       active: local # 기본 환경을 prod로 설정 local 개발할때는 local로 바주세요.
9
10  jwt:
11    key: service-key
12    live:
13      atk : 12096000000 # 2WN
14      rtk : 12096000000 #2주

```

## 4. Gradle을 이용해 빌드(jar파일 생성)

IntelliJ의 Gradle을 콘솔 빌드



## 5. docker-compose로 container띄우기

```

# ~/SSAFY/common_project/clone/S08P12D101/Server/ [develop-be*] docker-compose up -d
Creating network "server_default_bridge" with the default driver
Creating redis_boot ... done
Creating mongo ... done
Creating database ... done
Creating server_application_1 ... done
Creating server_batch-server_1 ... done
# ~/SSAFY/common_project/clone/S08P12D101/Server/ [develop-be*] docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS                               NAMES
1151c3c57e3a        server_application   "java '-Duser.timezone='..."   3 seconds ago     Up 2 seconds      0.0.0.0:8003->8003/tcp, 0.0.0.0:8003->8003/tcp   server_application_1
3f8530afdf788       server_batch-server  "java '-Duser.timezone='..."   3 seconds ago     Up 2 seconds      0.0.0.0:8002->8002/tcp, 0.0.0.0:8002->8002/tcp   server_batch-server_1
6febbedf20ec        redis:alpine        "docker-entrypoint.s..."   5 seconds ago     Up 3 seconds      0.0.0.0:6379->6379/tcp, 0.0.0.0:6379->6379/tcp   redis_boot
5ad3d14bc70e        mariadb             "docker-entrypoint.s..."   5 seconds ago     Up 3 seconds      0.0.0.0:3306->3306/tcp, 0.0.0.0:3306->3306/tcp   database
5a62c67df135        mongo:4.4.3        "docker-entrypoint.s..."   5 seconds ago     Up 3 seconds      0.0.0.0:27017->27017/tcp, 0.0.0.0:27017->27017/tcp   mongo

```

## 6. docker-compose로 container띄우기

접속이 잘되는지 테스트해보기

# 안드로이드



## 환경 구성

- 운영체제 : Window 10
- 준비물 : Android Studio, Git



## 빌드 및 테스트 방법

### 1. Repository clone

```
git clone --single-branch --branch develop-be https://lab.ssafy.com/s08-blockchain-contract-sub2/S08P22D208.git
```

### 2. app module: build.gradle

```
~/S08P22D208\Android\app
```

해당 파일에 web3j사용을 위한 라이브러리를 확인한다.

```
97
98      // web3j for wallet
99      implementation libs.web3j.core
100     implementation libs.web3j.contracts
101     implementation libs.bouncycastle
102
```

### 4. libs.versions.toml

```
web3j_core = "4.9.7"
web3j_contracts = "4.9.7"
bouncycastle = "1.68"
```

```
web3j-core = { module = "org.web3j:core", version.ref = "web3j_core" }
web3j-contracts = { module = "org.web3j:contracts", version.ref = "web3j_contracts" }
bouncycastle = { module = "org.bouncycastle:bcprov-jdk15on", version.ref = "bouncycastle" }
```

### 5. import 구문을 통해 web3j 라이브러리 활용 확인

```
import org.web3j.crypto.Credentials
import org.web3j.crypto.ECKeyPair
import org.web3j.crypto.Keys
import org.web3j.protocol.core.RemoteFunctionCall
```

```
import org.web3j.protocol.Web3j
import org.web3j.protocol.http.HttpService
import org.web3j.tx.ReadonlyTransactionManager
import org.web3j.tx.gas.StaticGasProvider
```



## 5. Kakao API 설정

### ▼ 카카오 로그인

Kakao Developers  
카카오 API를 활용하여 다양한 어플리케이션을 개발해보세요. 카카오 로그인, 메시지 보내기, 친구 API, 인공지능 API 등을 제공합니다.

 <https://developers.kakao.com/>

kakao developers

### 1. 카카오 Developers에서 애플리케이션 추가

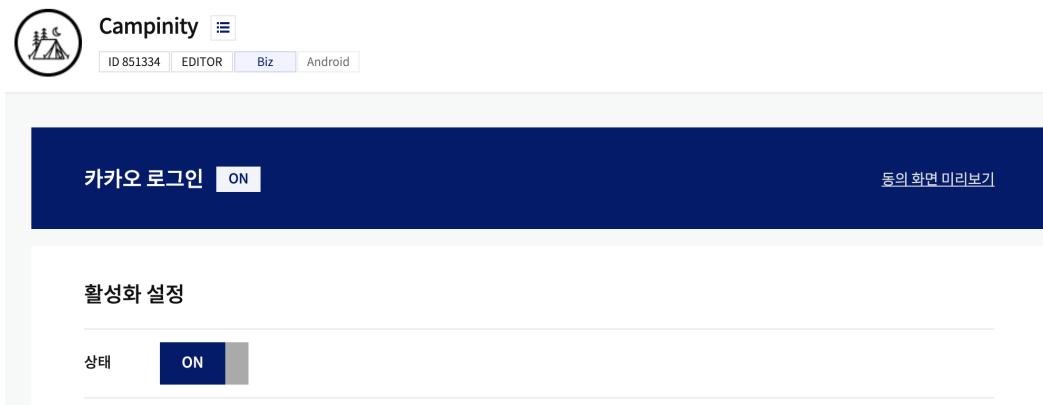
#### 애플리케이션 추가하기

앱 아이콘	<input type="button" value="이미지 업로드"/> 파일 선택 JPG, GIF, PNG 권장 사이즈 128px, 최대 250KB
앱 이름	내 애플리케이션 이름
사업자명	사업자 정보와 동일한 이름

- 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.

### 2. 카카오 로그인 활성화



### 3. Redirect URI 등록

#### Redirect URI

##### Redirect URI

카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다.

여러개의 URI를 줄바꿈으로 추가해주세요. (최대 10개)

REST API로 개발하는 경우 필수로 설정해야 합니다.

예시: (O) <https://example.com/oauth> (X) <https://www.example.com/oauth>

```
http://localhost:8003/api/v4/members/login-kakao  
http://i8d101.p.ssafy.io:8003/api/v4/members/login-kakao
```

취소

저장

## ▼ 부트페이 연동 방법

- 관리자 페이지 접속해서 rest api key 및 private key 확인

**1. boot pay 관리자 접속**

연동기 및 보안

결제연동을 위해 인증키를 확인 후 소스코드에 적용합니다.

**2. PG 설정**

결제설정

- 결제 수단 설정
- 동등기 및 보안
- 풀통 살정
- 서비스 설정

결제내역

주문결제

자동 결제 관리

데이터 분석

**연동기**

플랫폼	키 (Application ID)	설명
Javascript 키	6426738a755e27001ead619e	각 플랫폼 SDK에서 API 호출시에 사용됩니다.
Android 키	6426738a755e27001ead619f	각 플랫폼 SDK에서 API 호출시에 사용됩니다.
iOS 키	6426738a755e27001ead61a0	각 플랫폼 SDK에서 API 호출시에 사용됩니다.
<b>REST API 키</b>	<b>6426738a755e27001ead61a1</b>	REST API 호출 시 사용됩니다.

**3. 서버는 rest api key를 사용**

Private Key **RiFFkHvn49IC/Bz9+Qc04gVI8qqBR1eDA2NKMzAb98M=**

Feedback URL, Feedback Data Content Type 등 결제결제 설정은 [결제설정](#)에서 선택 해주세요.

**4. 위 private key 사용**

Feedback URL, Feedback Data Content Type 등 결제결제 설정은 [결제설정](#)에서 선택 해주세요.

IP 보안 OFF

IP 보안이 깨었습니다  
IP 보안이 깨진 경우 모든 서버의 요청이 수락됩니다.

환경설정

## 2. application.yml에 추가

```

ethereum:
  contract:
    wallet-address: '0xfB9843b34f1aB19d82Ba25DB6865897fA1311a74'
    song-address: ''
    nft-address: '0x2EcAE23EaE157fC279C2cb457624F112F47B387A'
    rpc-url: 'https://rpc.ssafy-blockchain.com'
    admin-address: '0x1e5ece0c6abecb6f0328734651c337ab3a524ddb'
    private-key: '26e3f26438ad12eb31278ae6d1db81d2a17d6fc2a24a48f06c3a8ec5dc50b8
    exchange-rate: 1000000

boot-pay:
  verification:
    rest-application-id: '6426738a755e27001ead61a1'
    private-key: 'RiFFkHvn49IC/Bz9+Qc04gVI8qqBR1eDA2NKMzAb98M='
  
```



## 6. 안드로이드 앱 apk 빌드 및 테스트 방법



### 환경 구성

- 운영체제 : Window 10(MAC Monterey도 동일)
- 준비물 : Android Studio, Git

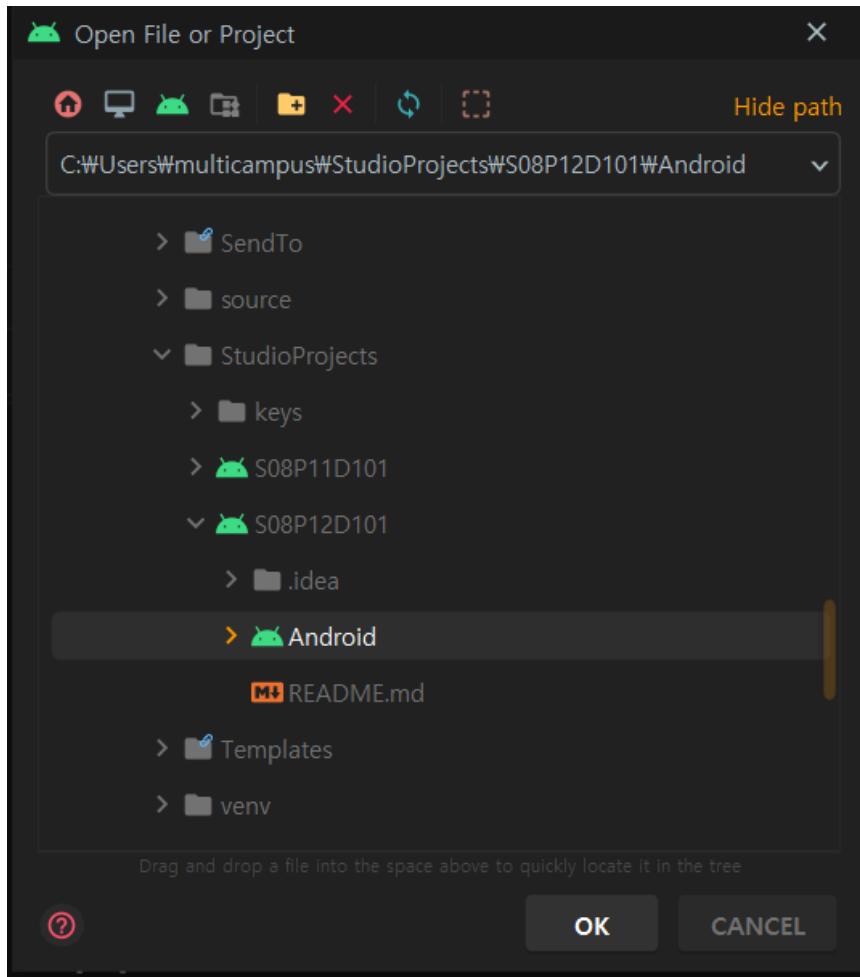


### 빌드 및 테스트 방법

#### 1. Repository clone

```
git clone --single-branch --branch develop-fe https://lab.ssafy.com/s08-blockchain-contract-sub2/S08P22D208.git
```

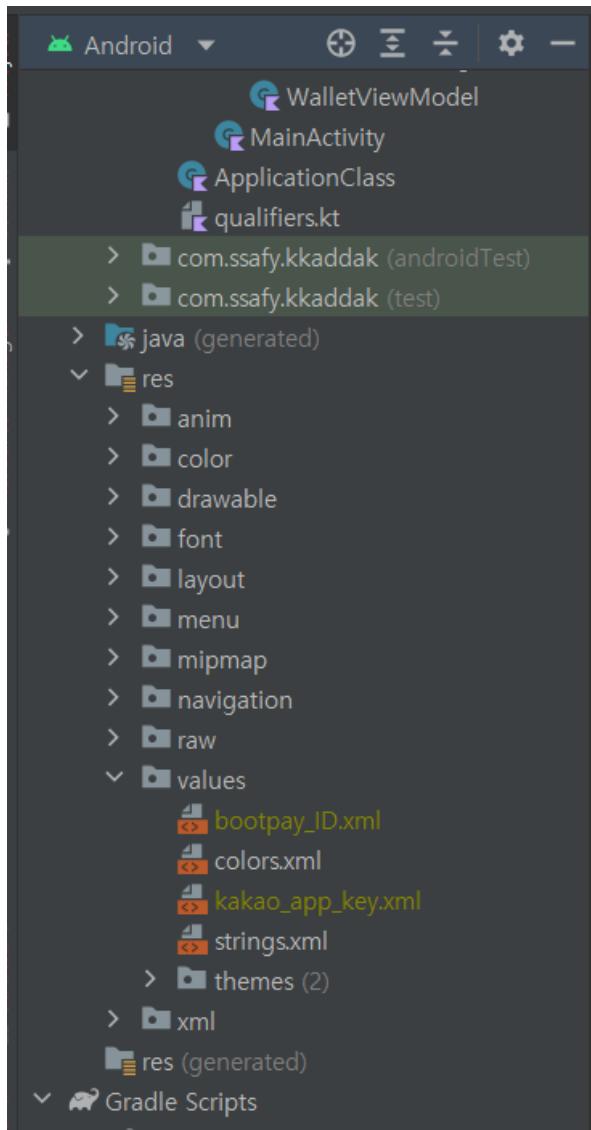
#### 2. 안드로이드 스튜디오 열기



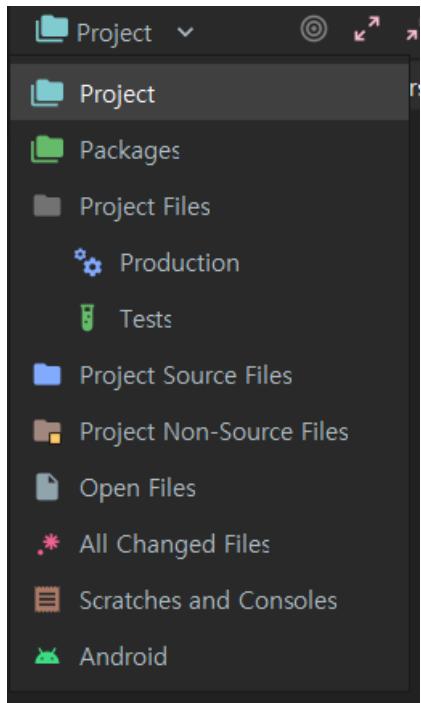
### 3. res의 values 폴더 클릭 후 kakao\_api\_key.xml 파일, bootpay\_ID.xml 불여넣기

[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f789dd2b-2b5d-42d1-9184-a21d7604ee33/kakao\\_app\\_key.xml](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f789dd2b-2b5d-42d1-9184-a21d7604ee33/kakao_app_key.xml)

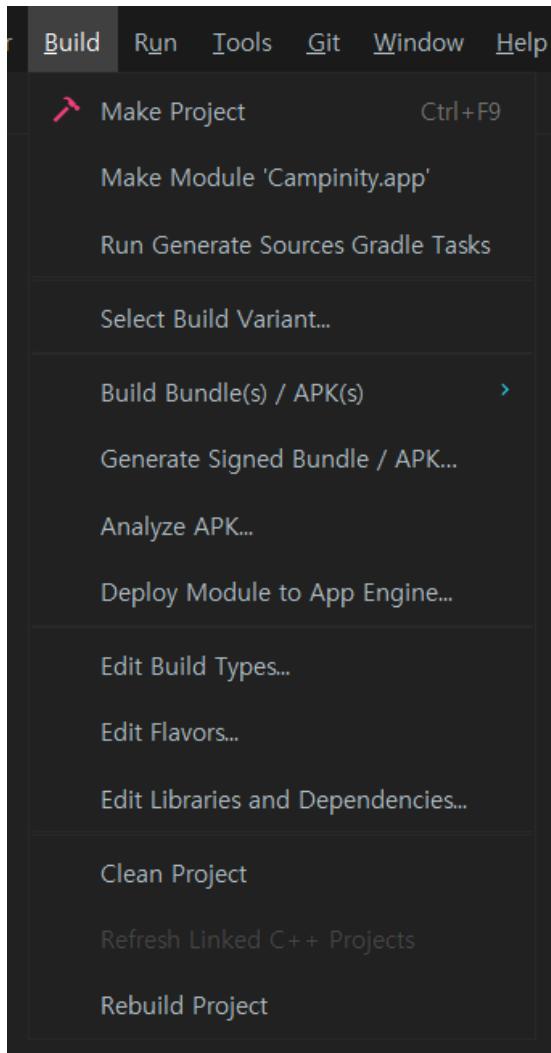
[https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cae93bb5-b7d8-4dbe-90bc-e2fe4f8ddd9d/bootpay\\_ID.xml](https://s3-us-west-2.amazonaws.com/secure.notion-static.com/cae93bb5-b7d8-4dbe-90bc-e2fe4f8ddd9d/bootpay_ID.xml)



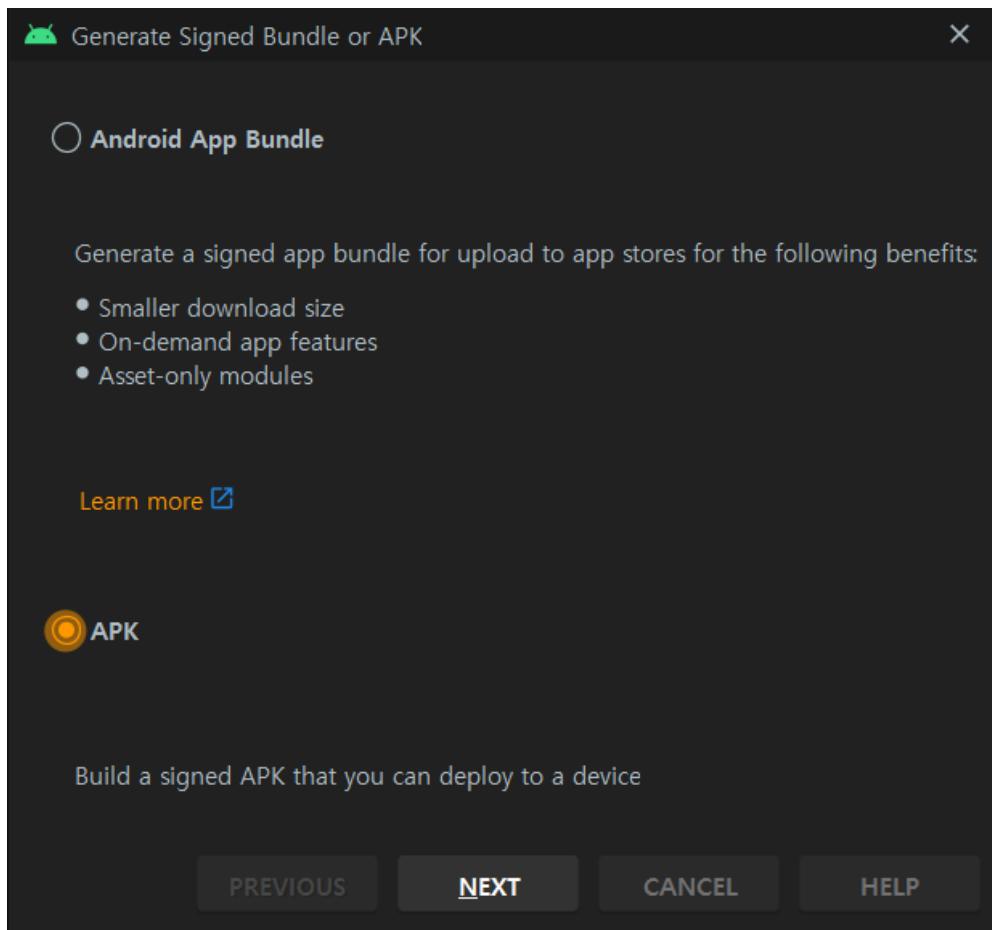
#### 4. 왼쪽 파일 계층 구조의 상단의 Android 클릭 후 Project로 계층 보기 변경



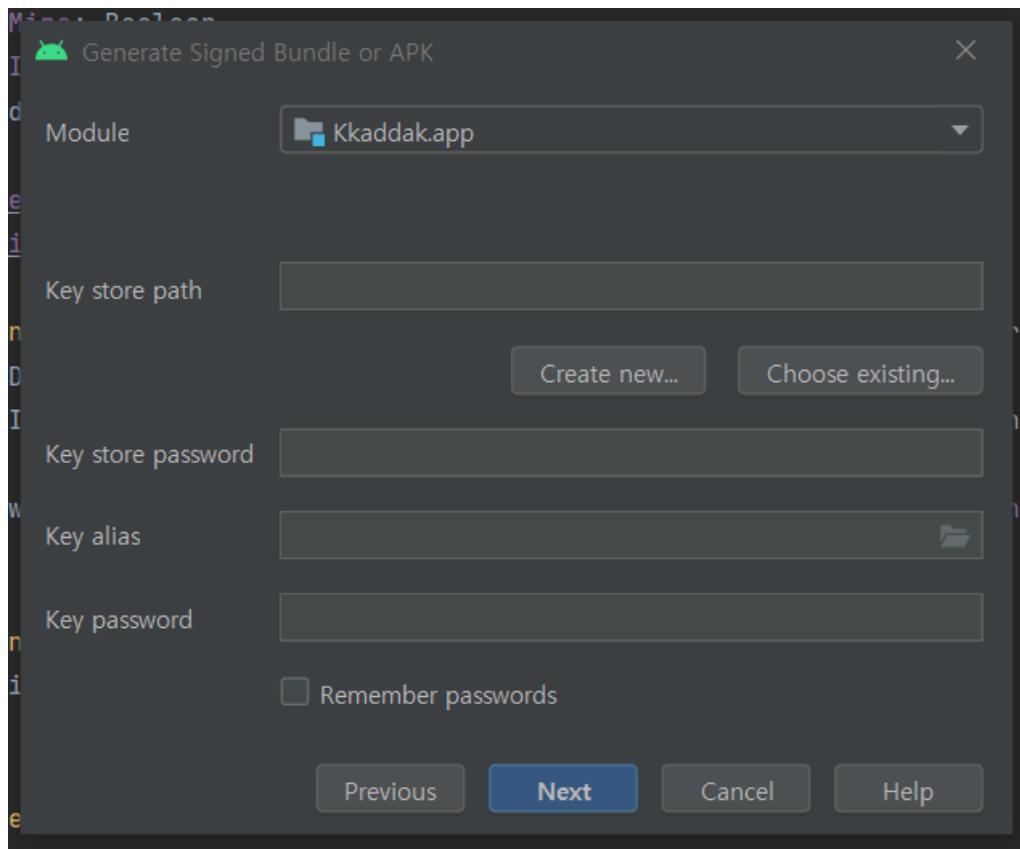
## 5. 상단 빌드 탭의 Generate Signed Bundle / APK 클릭



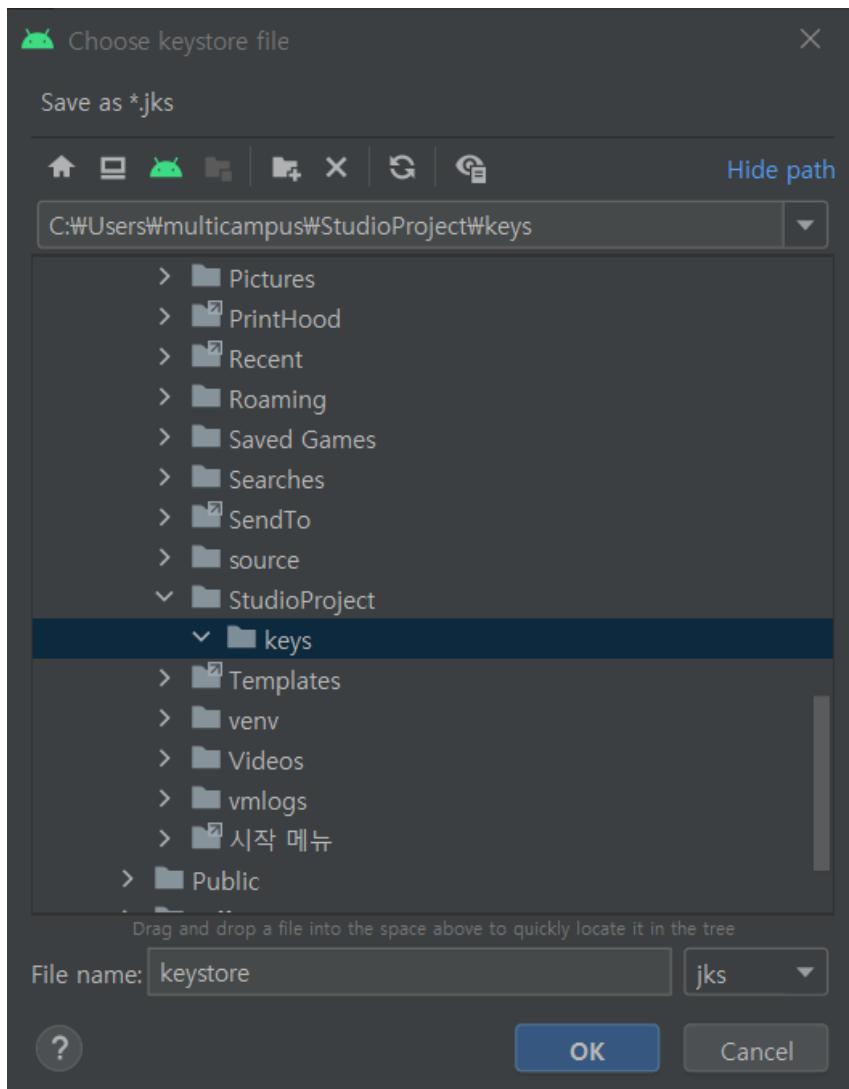
## 7. APK 선택, NEXT 클릭



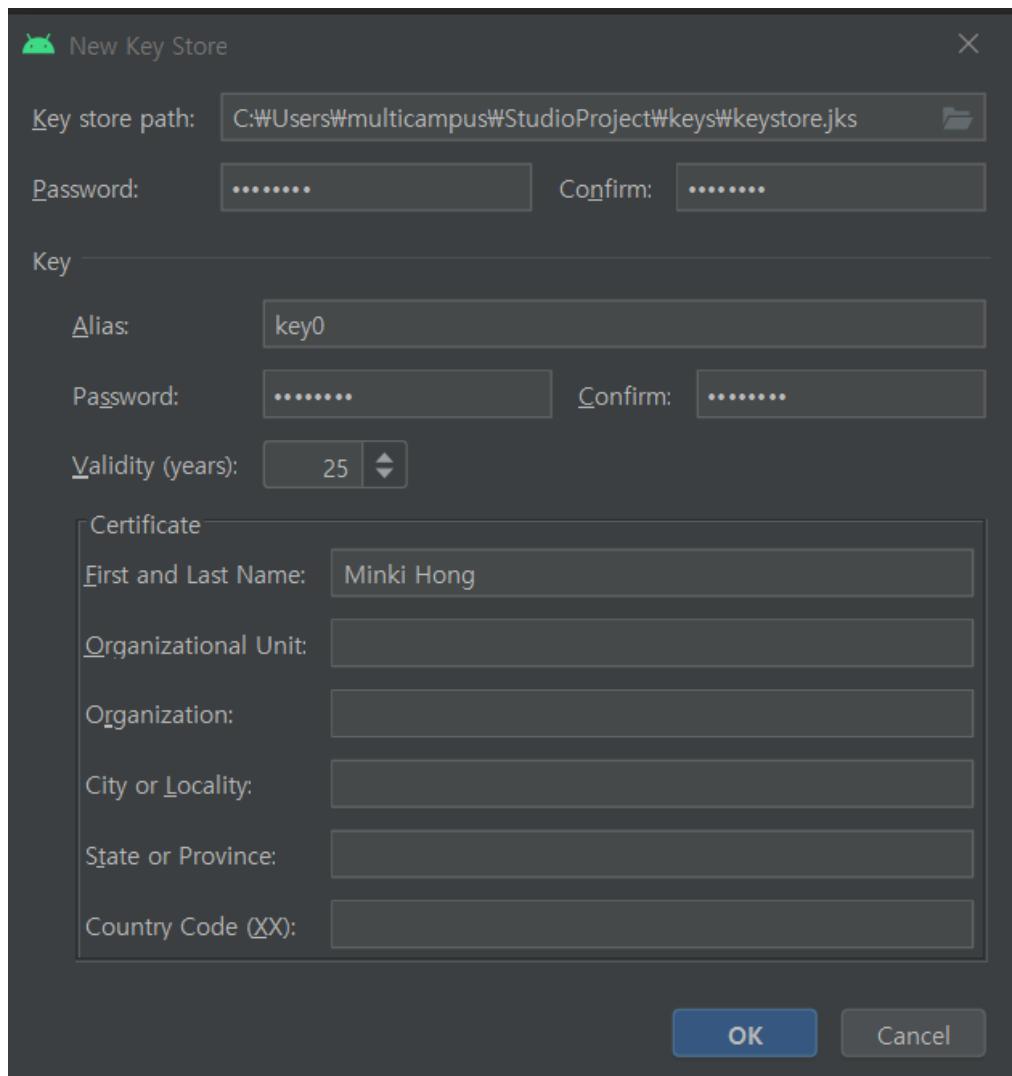
## 8. Key store path의 CREATE NEW 클릭



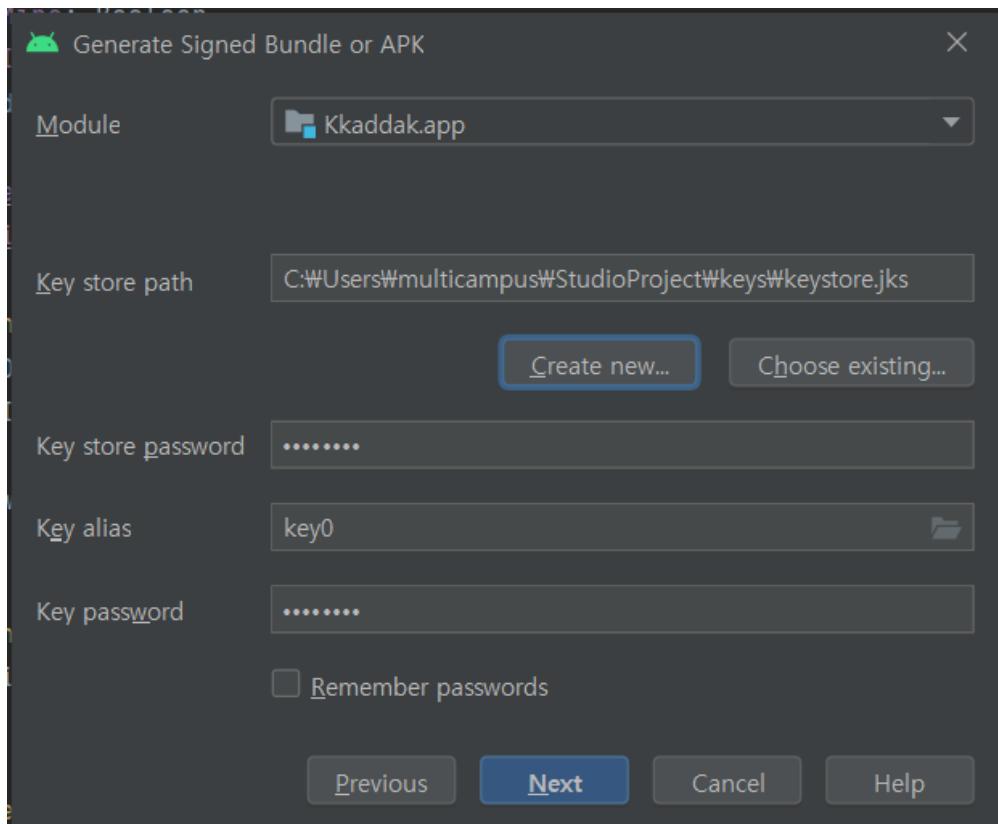
## 9. 저장 위치와 이름 설정 후 OK 클릭



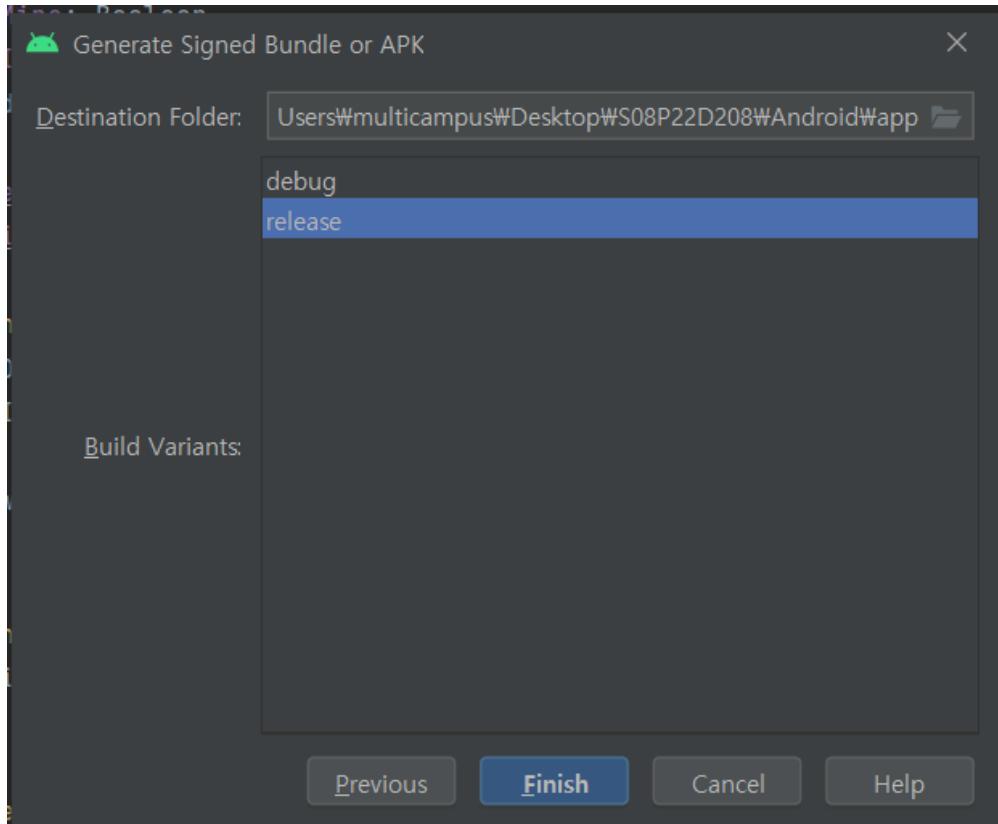
#### 10. 나머지 정보 입력 후 OK 클릭



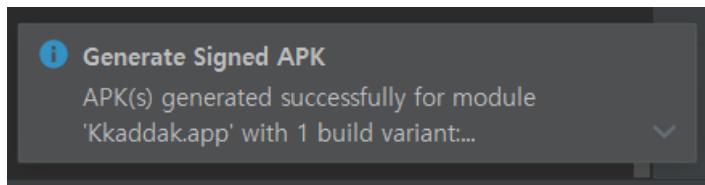
## 11. Next 클릭



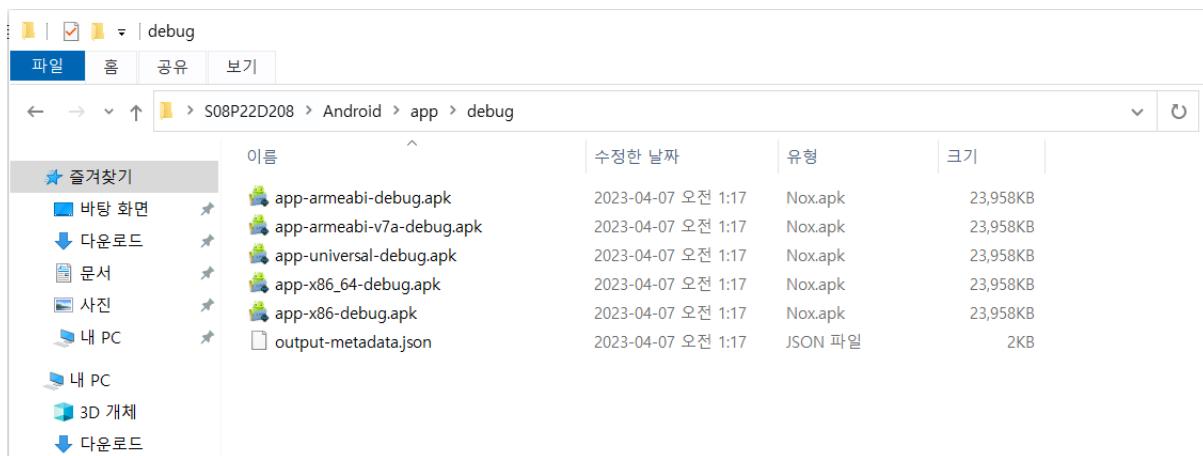
## 12. debug로 선택 후 FINISH 클릭



13. 빌드가 완료 된 후 하단에 나타난 popup을 확장하면 빌드된 파일이 있는 locate로 바로 이동 할 수 있다.

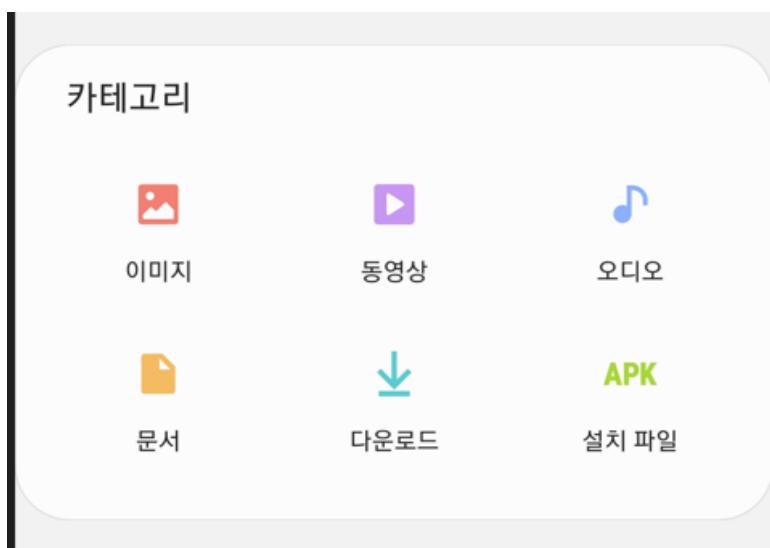


14. 빌드된 파일은 해당 프로젝트의 Android/app/release 폴더 내부에 위치하며, app-x86-debug.apk 파일을 이용한다.

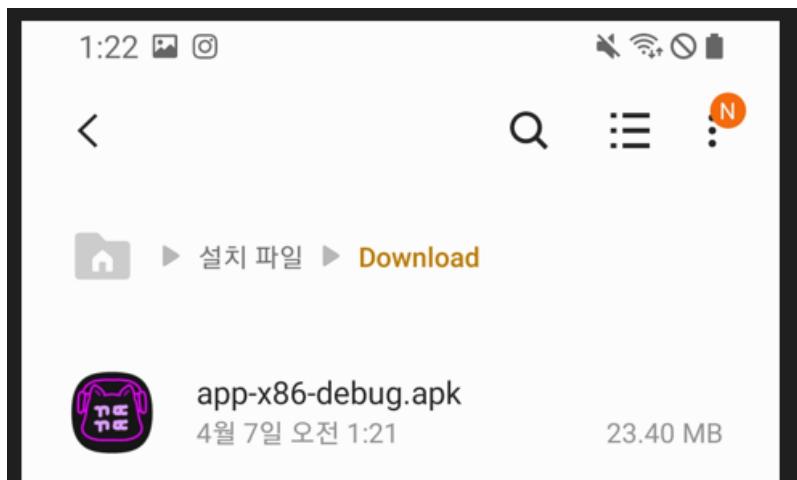


15. apk파일을 핸드폰으로 옮긴 후 설치한다.(개인 메신저나 usb 파일 공유 이용)

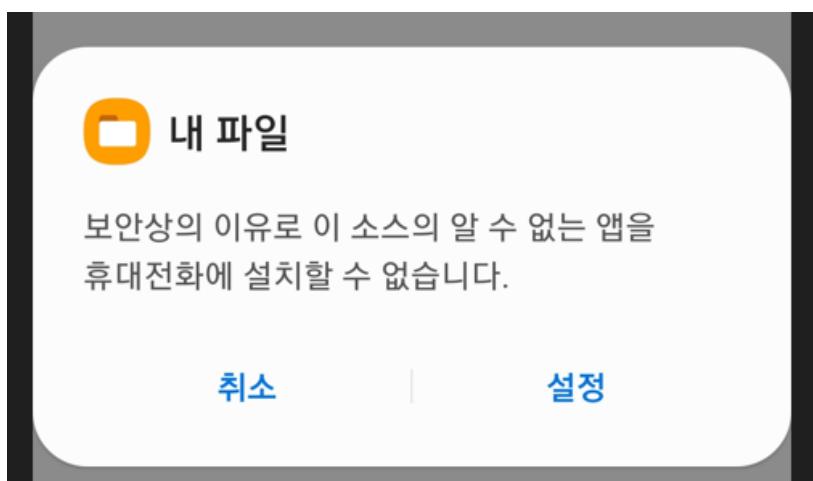
1. 폴더의 설치파일 카테고리 선택



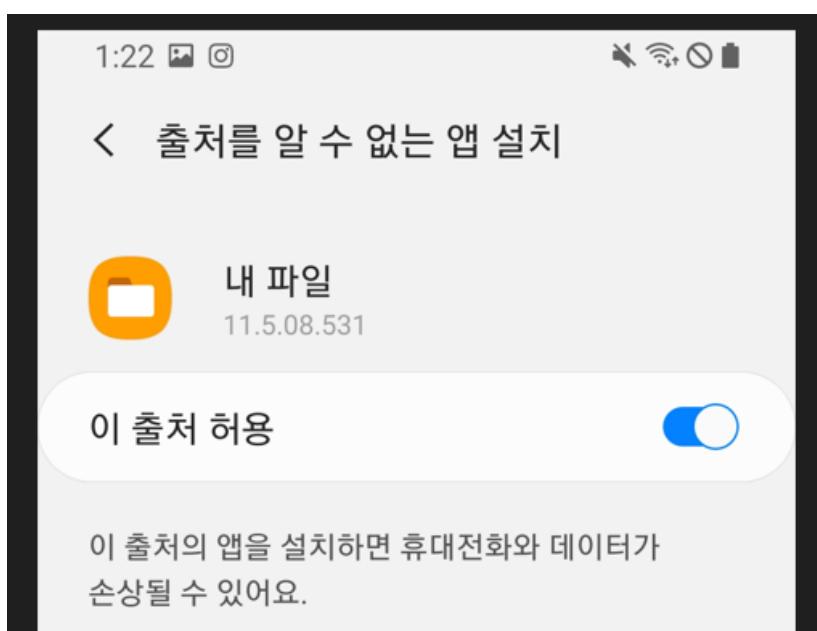
2. 설치할 apk 클릭



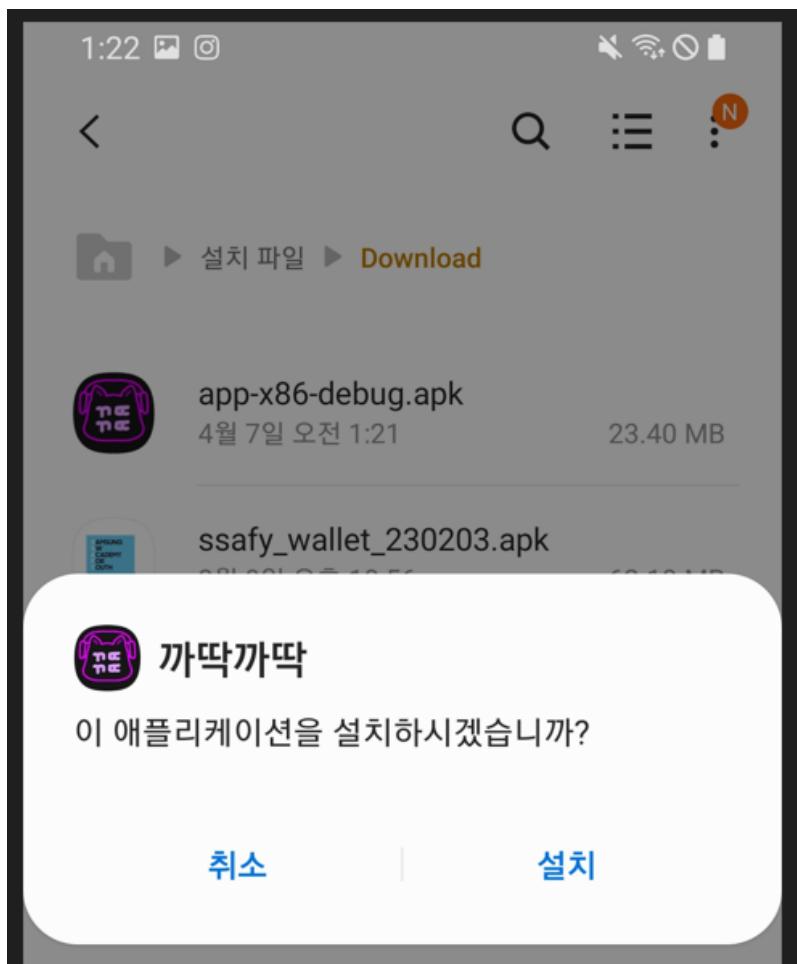
1. 출처를 알 수 없는 앱 설치 설정 변경



4. 내 파일의 설치를 허용(토글 버튼)



## 5. 앱 설치 클릭



## 6. 설치 완료





## 7. 시연 시나리오

### 홈

- 홈 화면에서는 창작자 정보와 대표 NFT, 그리고 신곡과 조회수에 따른 실시간 차트를 구현했습니다.
- 이렇게 음악 아이템을 클릭하게 되면 음악이 재생됩니다. 재생되는 음악은 즉시 플레이리스트에 추가됩니다.
- 현재 음악 상세화면을 보시면 음악이 1분만 제공되고 있습니다. 구독시 전체듣기를 제공함을 이렇게 도움말로 표시하고 있습니다.
- 실제로 구독을 하고 다시 음악을 재생해보겠습니다.
- 창작자 프로필로 가서 구독을 해보겠습니다.
- 구독에는 저희가 산정한 자체 코인 5까딱이 소요됩니다. 현재 잔액이 5까딱보다 많음으로 바로 결제가 가능합니다. 잔액이 부족할 경우 지갑페이지로 넘어가도록 구현해놓아 충전을 유도했습니다.
- 결제를 하고, 다시 음악을 재생해보면 (음악 재생)
- 이렇게 음악 시간이 3분 47초로 늘어난 것을 확인하실 수 있습니다.
- 또한, 음악 재생은 앱 내 모든 화면에서 유지되어 앱 내의 기능을 음악을 들으면서 사용 할 수 있습니다.

### 음원 페이지

- 음원 페이지에서는 등록된 음원에 대해 검색과 필터링을 구현했습니다

### 지갑 페이지

- 지갑 페이지에서는 자체 구현한 지갑을 통해 계정을 생성하거나 기존 계정을 불러올 수 있습니다. 이 화면에서는 잔액 정보를 확인할 수 있고 스마트 컨트랙트 거래 내역을 확인할 수 있습니다.
- 또한 신용카드와 간편 결제를 통해 앱 내 자체 코인을 충전할 수 있습니다.

- 실제로 500원의 원화를 까딱 코인으로 환전해보겠습니다. 결제 금액을 입력하고 카카오페이지로 결제를 해보겠습니다. 결제 후 충전까지는 약 1분 정도가 소요됩니다.

## 프로필 페이지

- 프로필 페이지는 저희 앱 내의 작은 커뮤니티 기능을 하고 있습니다. 본인의 프로필에서 는 업로드한 음악에 대한 유사도 검사 진행 상태를 확인할 수 있습니다. 화면에 보이는 reject 배지를 단 음원은 유사도 검사에서 불합격한 음원입니다. 이런 음원에 대해서는 삭제만 가능하도록 구현했습니다.
- 직접 음원을 올려보도록 하겠습니다.
- 제목과 음악 파일, 커버 사진, 그리고 장르와 분위기를 입력하면 음악을 등록하실 수 있습니다.
- 음악이 올라가고 실시간으로 유사도 검사 상태가 반영됩니다. 유사도 검사가 approve 되었습니다. 아이템을 클릭해보면 nft를 발급받을 수 있다는 알림창이 나오게 됩니다. 확인을 누르는 동시에 음원에 대한 nft가 발급되고, 음악도 블록체인에 등록됩니다. 이렇게 승인 상태에서 확인을 눌러 완료 상태가 된 음원은 다른 유저들도 프로필 페이지에서 확인할 수 있고, 홈 화면이나 음악 리스트 화면에서도 확인할 수 있게 됩니다.
- 그러면, 등록된 음원에 대한 NFT를 확인해보도록 하겠습니다. 이렇게 NFT가 발급되었습니다. NFT는 배경에는 아까 등록한 커버사진이 블러처리되어 보여지게 됩니다. 약 12만 가지의 경우의 수에 배경사진까지 경우를 더하여 NFT의 유일성을 보장할 수 있습니다.
- NFT 상세화면에서 갤러리로 NFT 사진을 저장할 수도 있고 인스타그램 스토리로 바로 공유도 가능합니다. 다시 돌아와서 홈 프로필에 등록을 할 수도 있습니다.
- 이렇게 반영이 된 모습을 확인할 수 있습니다.

## 판매 페이지

- 다음은 NFT 마켓플레이스입니다. 음원을 올리고 발급받은 NFT를 판매할 수 있습니다. 실제로 판매를 해보도록 하겠습니다.
- 등록창에 가서 보유한 NFT 중 하나를 선택하면 노래 제목이 자동으로 입력됩니다. 간단하게 가격만 입력을 하고 판매가 가능합니다.
- 이렇게 판매 내용이 등록되었습니다.
- 판매 중인 NFT는 다른 사용자가 구매할 경우 판매가 종료되고, 판매를 멈추고 싶을 경우 직접 종료도 가능합니다.

- 다른 사용자의 NFT를 구매해보겠습니다. 이렇게 구매버튼을 누르고 다시 프로필 페이지로 가보면 구매한 NFT가 프로필에 등록되어 있는 것을 확인하실 수 있습니다.
- 또한, 지갑페이지에서 구매한 내역 정보를 확인하실 수 있습니다.

시연을 마치도록 하겠습니다.