



1. 개발 환경

형상 관리

- Gitlab

이슈 관리

- Jira

Communication

- Notion
- Discord
- Google Meet

UI/UX

- Figma

OS

- Window 10
- MacOS Monterey

IDE

- IntelliJ 2022 1.3
- Android Studio

Server

- AWS EC2
 - Ubuntu 20.04LTS
 - Docker 20.10.18

DataBase

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

기타 편의 툴

- PostMan 9.28.1



Front-End

Back-End









language

-  **JAVA** **11.0.15**  **PYTHON** **3.11.1**



framework

-  **SPRING BOOT** **2.7.9**
-  **FLASK** **1.1.4**

library

-  **SPRING DATA JPA** **2.7.9**  **SPRING BATCH** **4.3.3**
-  **SPRING SECURITY** **5.6.1**  **THYMELEAF** **3.0.12**
-  **FIREBASE** **9.1.1**  **WEBSOCKET** **2.6.2**
-  **JUNIT5** **5.7.0**  **SWAGGER** **3.0.0**

database

-  **MARIADB** **10.10.2**  **MONGODB** **4.4.3**
-  **REDIS** **7.0.8**  **H2 DATABASE** **2.7.9**

CI/CD & Containerization

-  **JENKINS** **LTS**  **DOCKER** **4.15.0**

build tools

-  **GRADLE** **8.0**



2. 배포서버 환경 구성(CI/CD)

목차

[방화벽 설치](#)
[Docker 및 docker-compose 설치](#)
[도커 실행](#)
[젠킨스 설치 및 실행](#)
[젠킨스 초기 설정](#)
[Jenkins 플러그인 추가 설치](#)
[젠킨스 Credential 등록](#)
[젠킨스 세부 설정](#)
[Gitlab WebHook 설정](#)
[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 GPG key 인증

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

5. Docker Repository등록

```
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 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 작성하셔야 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

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 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
```

▼ 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 다운로드된다.
# 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.
```

▼ 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

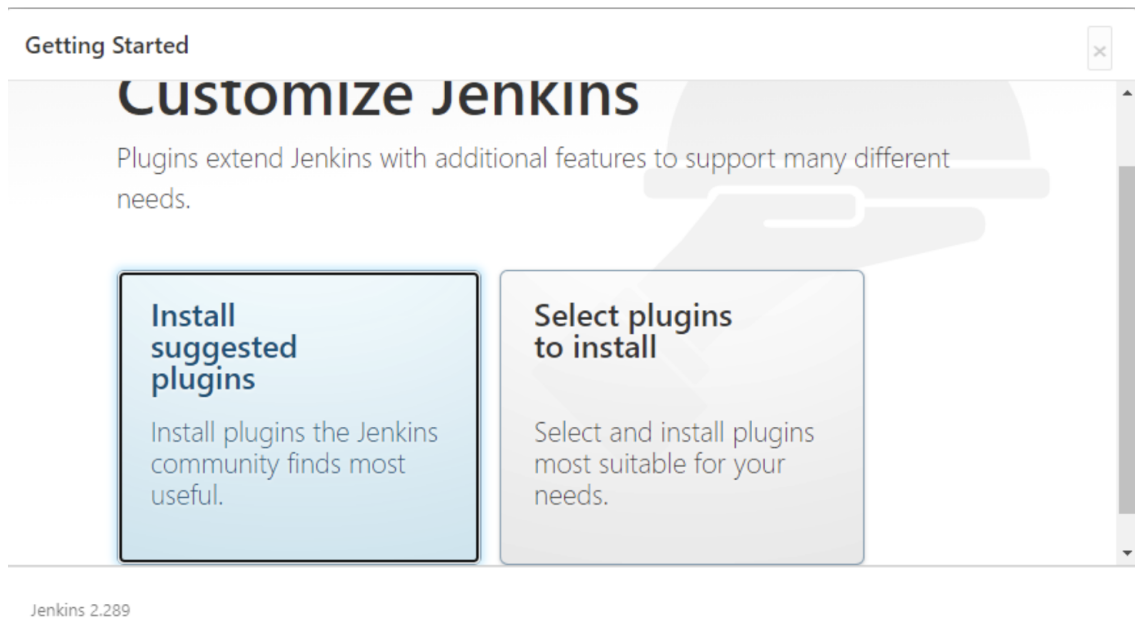
- 우선 jenkins 컨테이너에 접속한다

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

- 초기 관리자 비밀번호를 확인한다

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

- 비밀번호를 입력하면 아래와 같은 화면이 나오는데, Install suggested plugins 클릭



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

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	Workspace Cleanup 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
✓ Timestamp	✓ 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

Dashboard > Jenkins 관리 > Plugin Manager

Updates
 Available plugins
 Installed plugins
 Advanced settings

Plugins

Install	Name ↓	Released
---------	--------	----------

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. Jenkins관리 → Manage Credential에 들어간다

Security



Configure Global Security

Secure Jenkins; define who is allowed to access/use the system.



Manage Credentials

Configure credentials



Configure Credential Providers

Configure the credential providers and types



Manage Users

Create/delete/modify users that can log in to this Jenkins.

2. System 선택

Credentials

T	P	Store ↓	Domain
		System	(global)

3. 우측 상단에 Add Credentials 클릭

Global credentials (unrestricted)

[+ Add Credentials](#)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
ubermer5ch1308	ubermer5ch1308/***** (gitlab)	Username with password	gitlab

4. 우선 Gitlab 프로젝트 Repository에 들어가서 accesstoken을 발급받는다

Campinity

Project information

Repository

Issues 0

Merge requests 0

CI/CD

Security & Compliance

Deployments

Packages and registries

Infrastructure

Monitor

Analytics

Wiki

Snippets

Settings

General

Integrations

Webhooks

Access Tokens

s08-webmobile4-sub2 > Campinity > Access Tokens

Search page

Project Access Tokens

Generate project access tokens scoped to this project for your applications that need access to the GitLab API.
You can also use project access tokens with Git to authenticate over HTTP(S). [Learn more.](#)

Add a project access token

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

Token name

For example, the application using the token or the purpose of the token. Do not give sensitive information for the name of the token, as it will be visible to all project members.

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
ubermer5ch1308 **Gitlab username**

☐ Treat username as secret

Password
Project accesstoken입력

ID
Gitlab ID입력

Description

Create


▼ 젠킨스 세부 설정


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

Enter an item name

campinity-develop2

» 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.


Dashboard > ttrip-develop > Configuration


Configure


 **General**

 소스 코드 관리

 빌드 유발

 빌드 환경

 Build Steps

 빌드 후 조치

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

 Please enter Git repository.

Credentials ?

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

Branches to build ?

Branch Specifier (blank for 'any') ?

2. 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

3. Build Steps 설정은 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/ttrip-develop/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

▼ 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`

`http://i8d101.p.ssafy.io:8080/project/campinity-develop`

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

`deploy-be`

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.

☒ 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 [redacted] 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

▼ Jenkins 설정 후 서버 빌드 및 배포 방법

자동으로 빌드 및 배포

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

수동으로 빌드 및 배포

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 ▼

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

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.sql을 다운받아서 sql을 execute 해준다.



4. 빌드 및 테스트 방법

백엔드

1. Repository clone
2. core application.yml 수정
4. Gradle을 이용해 빌드(jar파일 생성)
5. docker-compose로 container 띄우기
6. api 확인

안드로이드

1. Repository clone
2. app module: build.gradle
4. libs.versions.toml

백엔드



환경 구성

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



빌드 및 테스트 방법

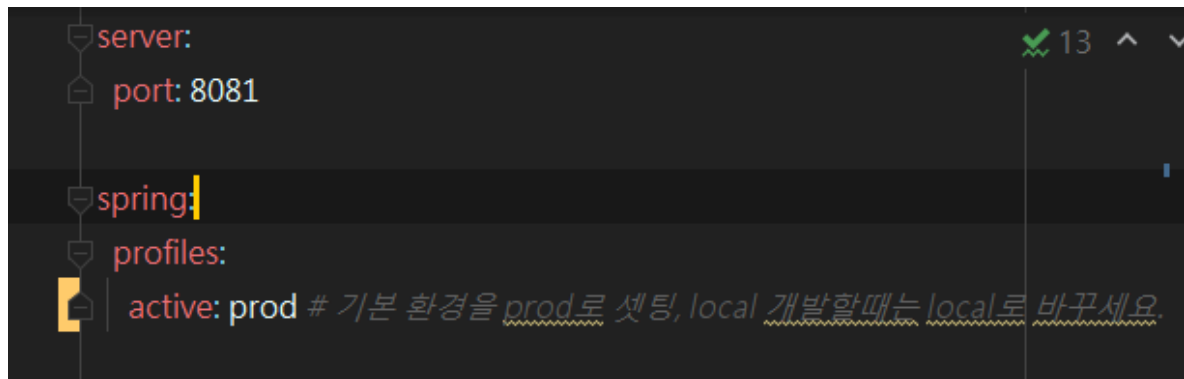
1. Repository clone

```
git clone https://lab.ssafy.com/s08-final/S08P31D104.git
```

2. core application.yml 수정

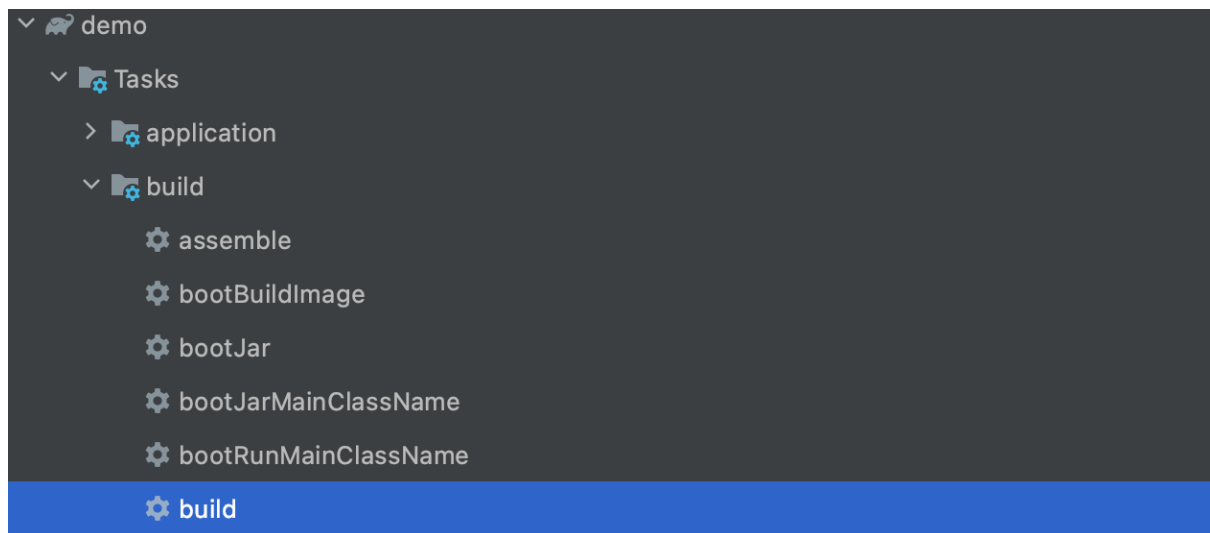
```
~/S08P31D104/Server/core/src/main/resources/application.yml
```

위 경로에 있는 파일을 열어 spring:profiles:active: prod로 수정한다



4. Gradle을 이용해 빌드(jar파일 생성)

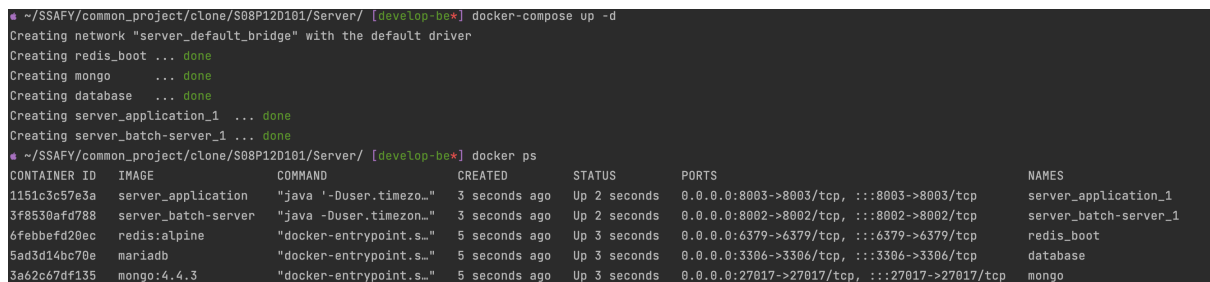
IntelliJ의 Gradle을 컨후 빌드



5. docker-compose로 container띄우기

~/S08P31D104/Server/에서 명령어 실행

```
docker-compose up -d --build
```



6. api 확인

도커 컨테이너들이 실행된 후 <http://localhost:8081/swagger-ui/index.html> 접속

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-final/S08P31D104.git
```

2. app module: build.gradle

```
~/S08P31D104\Android\app
```

해당 파일에 google map, firebase, arccore등 사용을 위한 라이브러리를 확인한다.

```

implementation libs.google.maps
implementation libs.google.maps.location
implementation libs.google.maps.directions

implementation libs.stomp.protocol

implementation libs.firebase.auth
implementation platform(libs.firebase.bom)
implementation libs.firebase.messaging

implementation libs.arcore
implementation libs.sceneform.ux
implementation libs.sceneform.core

```

4. libs.versions.toml

```

google-maps-services-maps = "18.0.2"

firebase = "31.4.0"

arcore = "1.35.0"

...

```

```

google-maps = { module = "com.google.android.gms:play-services-maps", version.ref = "google-maps-services-maps" }
google-maps-location = { module = "com.google.android.gms:play-services-location", version.ref = "google-maps-services-location" }
google-maps-directions = { module = "com.google.maps:google-maps-services", version.ref = "google-maps-services-directions" }
stomp-protocol = { module = "com.github.NaikSoftware:StompProtocolAndroid", version.ref = "stomp" }
firebase-bom = { module = "com.google.firebase:firebase-bom", version.ref = "firebase" }
firebase-auth = { module = "com.google.firebase:firebase-auth-ktx" }
firebase-messaging = { module = "com.google.firebase:firebase-messaging-ktx" }
openvidu = { module = "io.openvidu:openvidu-android", version.ref = "openvidu" }
arcore = { module = "com.google.ar:core", version.ref = "arcore" }
sceneform-ux = { module = "com.google.ar.sceneform.ux:sceneform-ux", version.ref = "sceneform" }
sceneform-core = { module = "com.google.ar.sceneform:core", version.ref = "sceneform" }

```



5. 안드로이드 앱 apk 빌드 및 테스트 방법



환경 구성

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

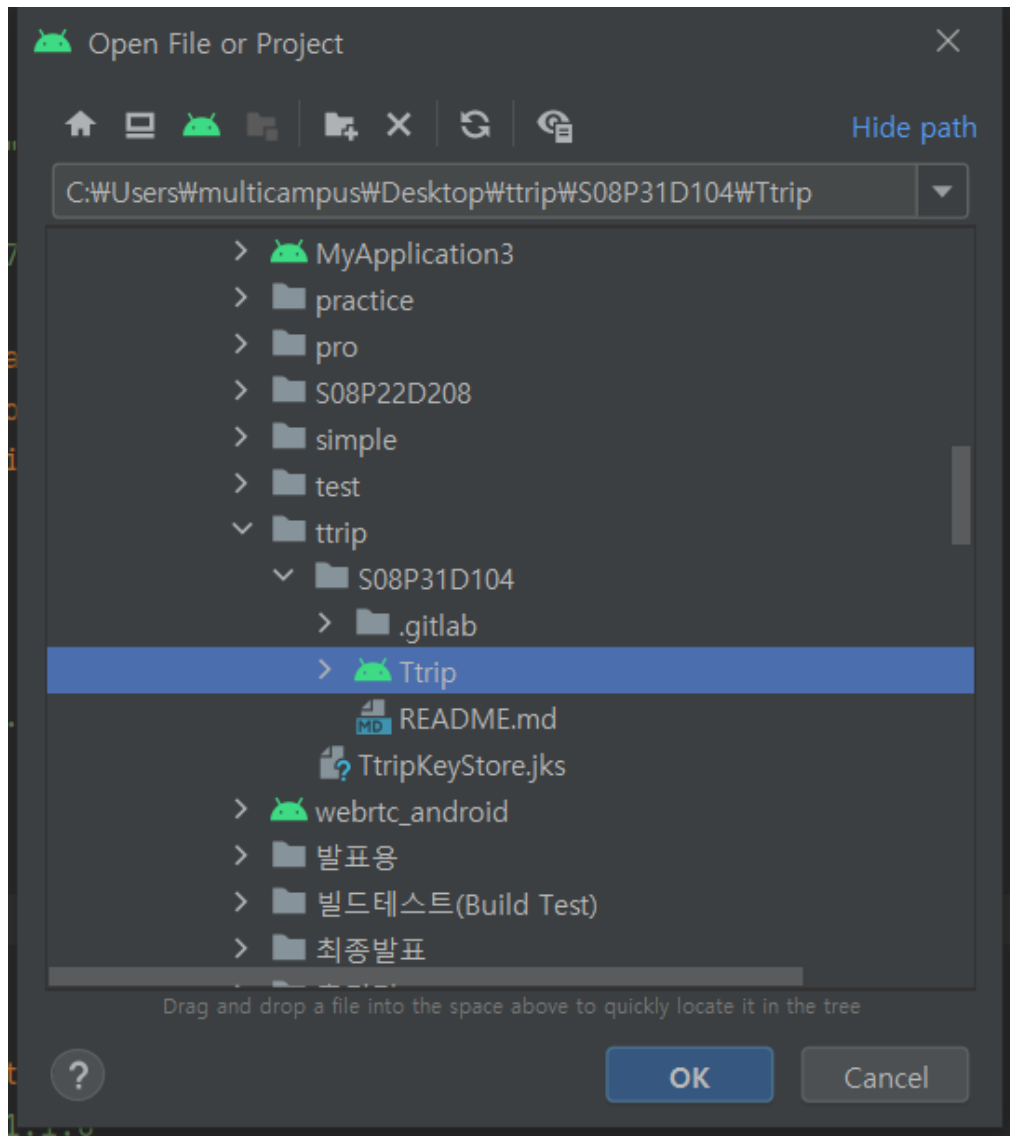


빌드 및 테스트 방법

1. Repository clone

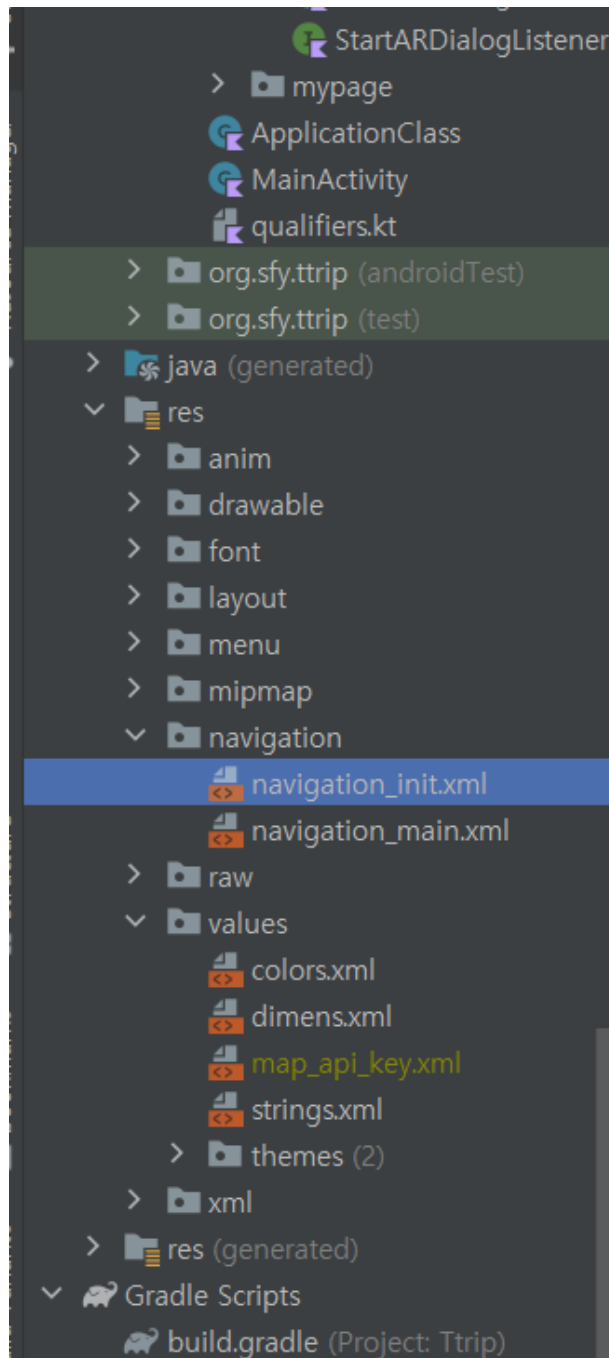
```
git clone --single-branch --branch develop-fe https://lab.ssafy.com/s08-final/S08P31D104.git
```

2. 안드로이드 스튜디오 열기



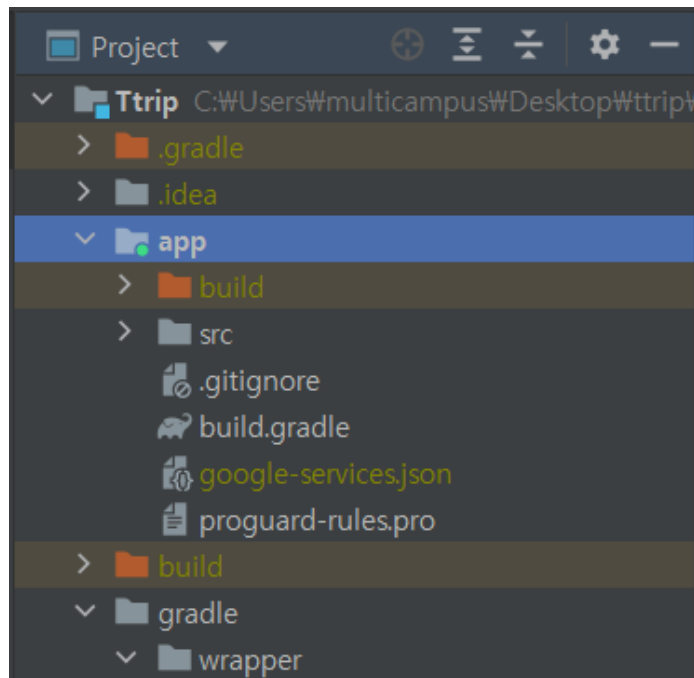
3. res의 values 폴더 클릭 후 map_api_key.xml 파일 붙여넣기

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4715b767-941d-4be9-9574-a6670c65bcc0/map_api_key.xml

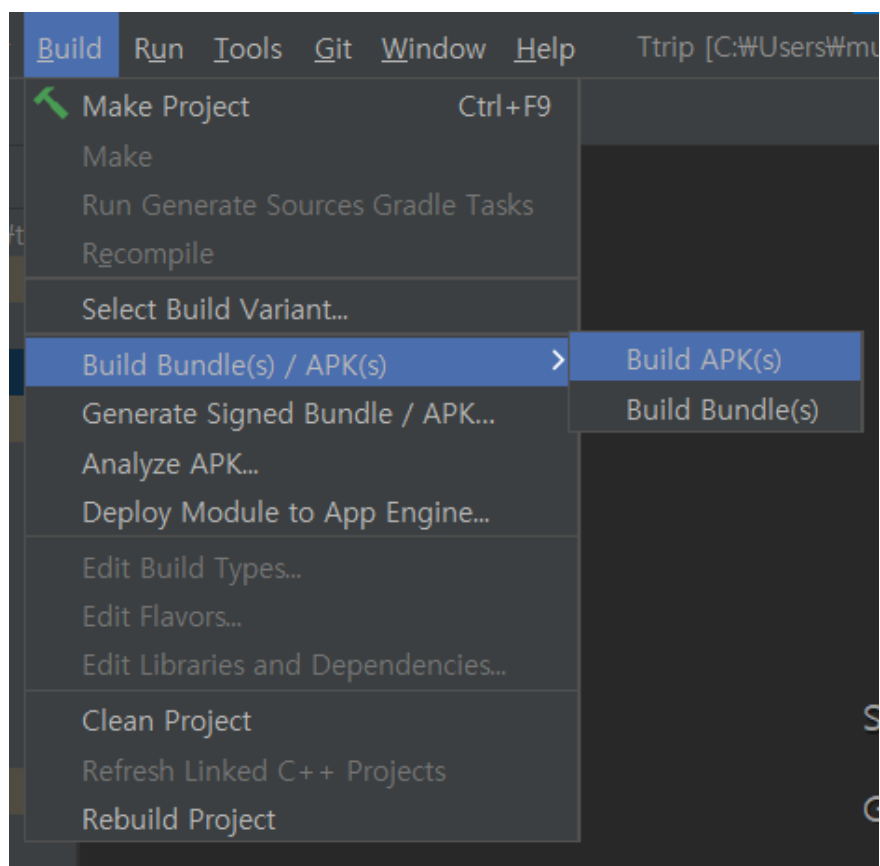


4. 왼쪽 파일 계층 구조의 상단의 Android 클릭 후 Project로 계층 보기 변경 후 app폴더에 아래 파일 추가

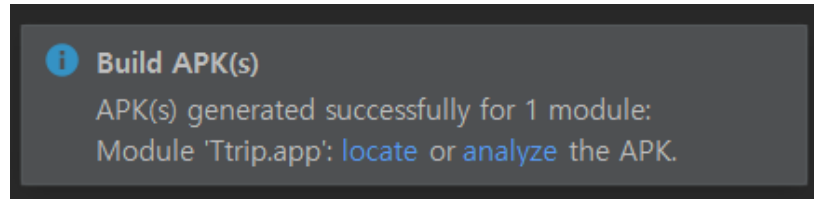
<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c6143977-f18a-47a9-8f28-9836afbc346/google-services.json>



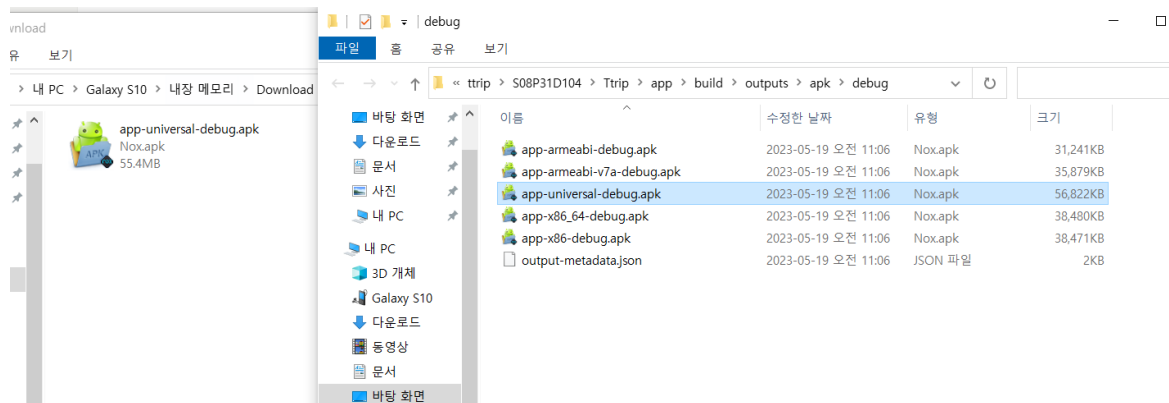
5. 상단 빌드 탭의 Build Bundle / APK(s) → Build APK(s) 클릭

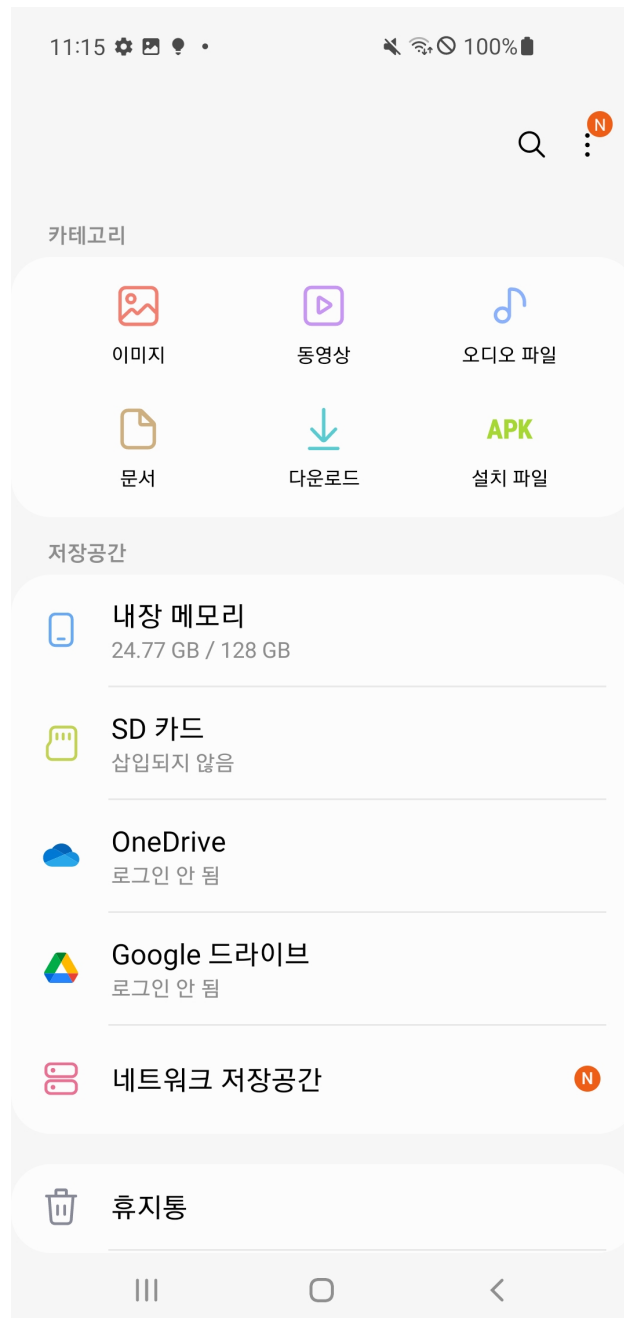


7. APK 빌드 완료 확인 안내

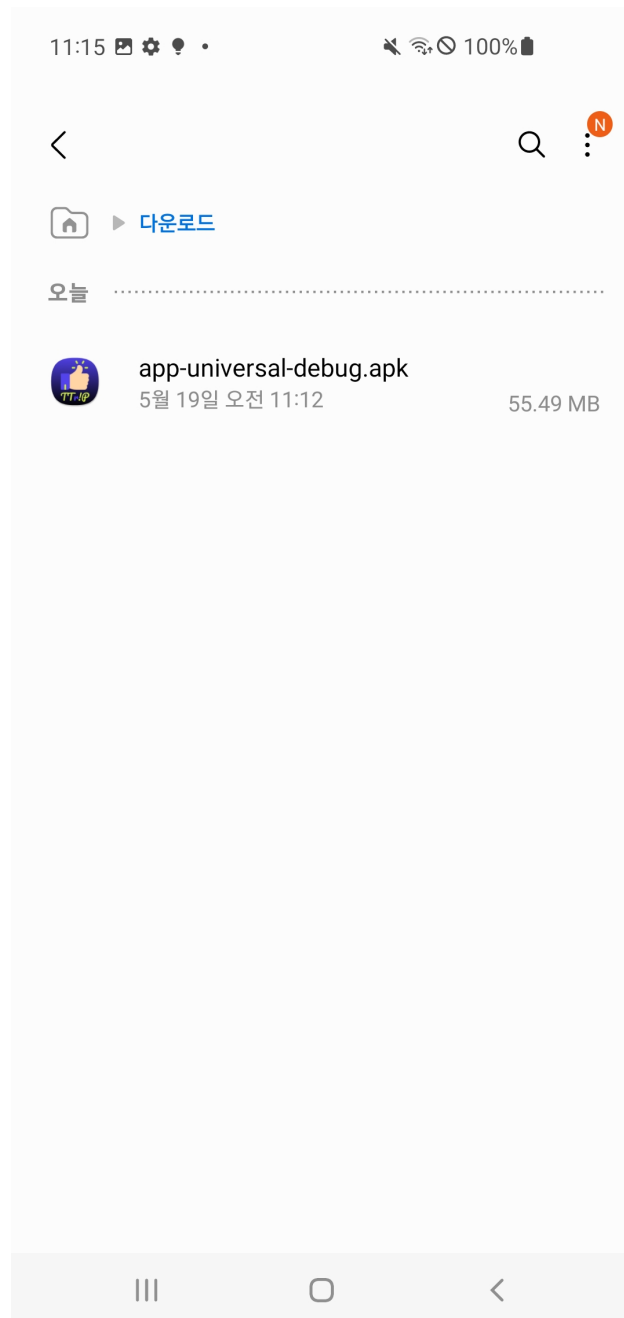


8. ~app\build\outputs\apk\debug 폴더에서 생성 확인 및 테스트 폰 다운로드 파일로 복사

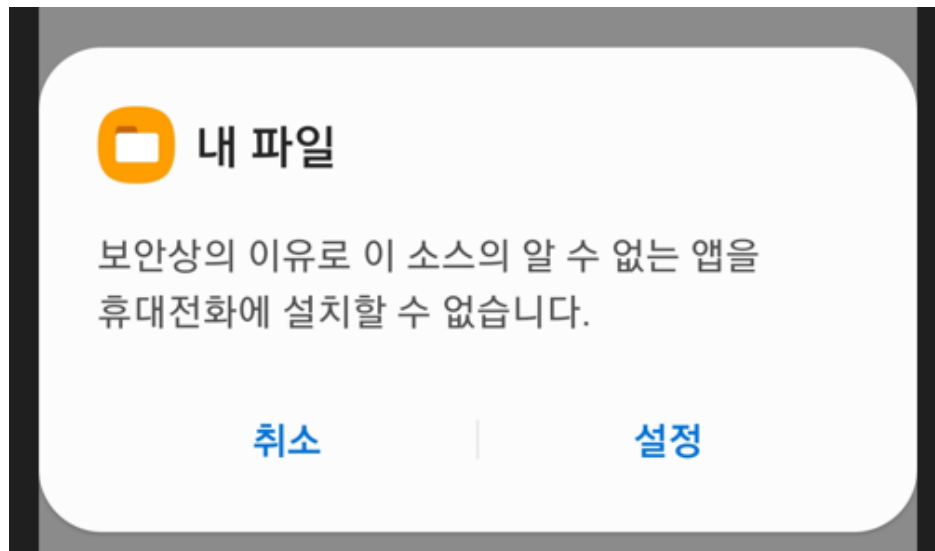




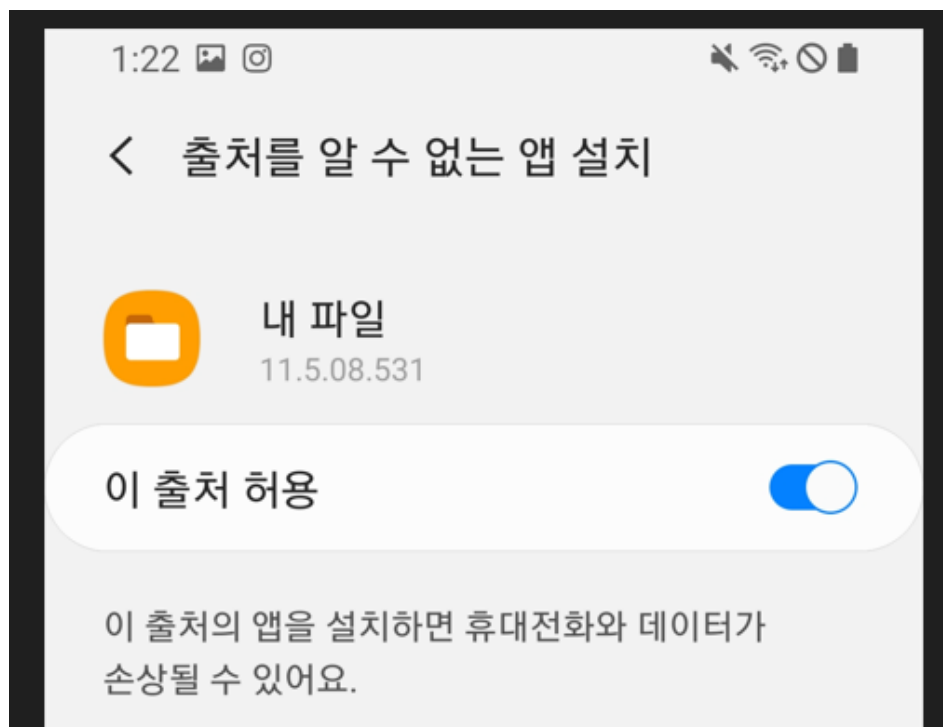
9. 복사된 apk 확인



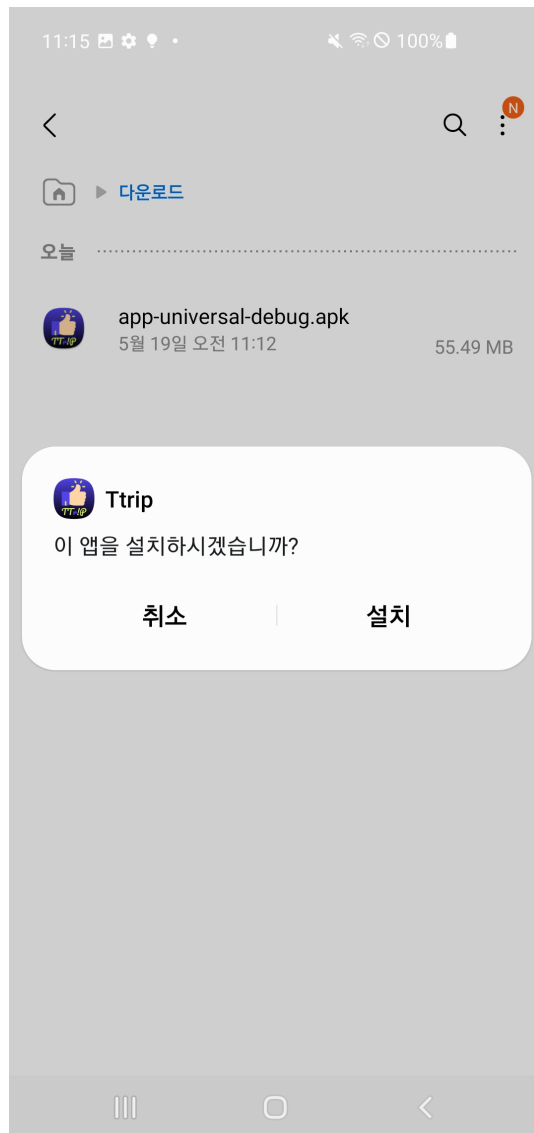
10. 설치 진행 시 출처를 알 수 없는 앱 설치 설정 변경



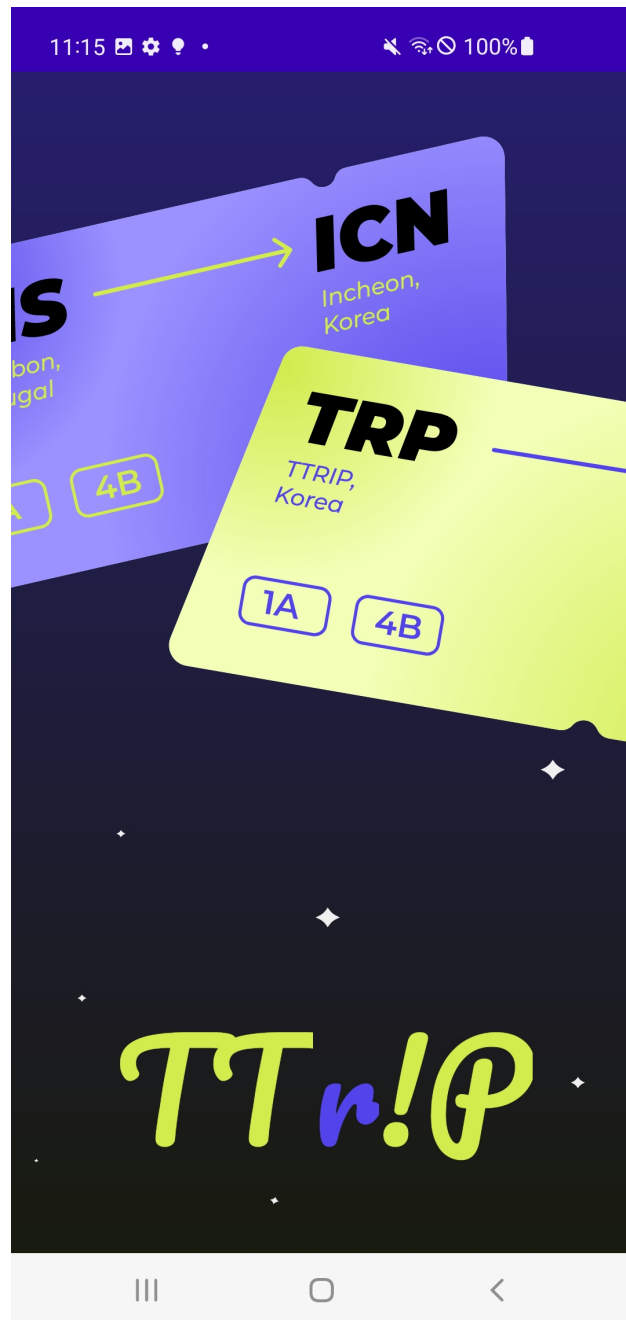
11. 내 파일의 설치를 허용(토글 버튼)



12. 앱 설치 진행



13. 앱 실행 완료





6.시연 시나리오

나레이션

지금부터 오늘은 유럽 여행 떠나기 전 날 밤입니다.

딤에 회원가입을 해보겠습니다.

시간상 기초적인 회원가입은 넘어가고, 취향조사만 해보겠습니다.

(취향 조사 페이지)

마치 Mbt처럼, 내 여행 취향을 선택해서 입력할 수 있습니다.

그럼 이 결과를 바탕으로 나와 여행 취향이 비슷한 유저를 확인할 수 있습니다.

홈화면입니다.

게시글들이 티켓 형태로 보입니다.

게시글은 기본적으로 최신순 정렬입니다.

그리고 마감이 가까운 순으로 빨강-초록-파랑으로 티켓 색을 나타냅니다.

즉 빨강이 제일 마감이 임박한 게시글이라는 뜻입니다.

직접 게시글을 작성해보겠습니다.

글의 제목을 입력하고

(26일 루브르 가실 분?)

내용을 입력합니다.

(같이 구경해요.)

여행지를 선택합니다.

(프랑스 파리)

여행 시작일은 내일 날짜인 5월 20일,

(5월 20일 설정)

여행 종료일은 한 달 뒤인 6월 20일로 하겠습니다.

(6월 20일로 설정)

자 게시글 등록을 완료했습니다.

그러면 방금 제가 올린 게시글과 유사한 게시글이 추천됩니다.

지금은 이 정도만 보고 넘어가겠습니다.

오늘은 여행 전 날이니 여기까지 하고, 이만 자도록 하겠습니다.

자 여행 당일 아침이 밝았습니다.

(팀원 두 명이 게시글에 지원)

루브르 동행을 구하는 제 게시글에 두 명이 지원했네요.

둘 중 저와 매칭률이 더 높은 림림씨에게 채팅을 걸어보겠습니다.

채팅:

재완: 안녕하세요

림림: 안녕하세요

이후 림림씨와 충분히 대화를 했다고 가정하고 매칭하겠습니다.

이렇게 동행 약속을 잡고, 저는 이만 파리행 비행기를 타러 갈게요.

파리에 도착했습니다.

오늘은 프랑스 맛집을 가볼까 합니다.

어... 제가 지금 인터넷에서 찾아봤는데 식당이 2인분 이상부터 주문 가능이네요.

그렇다면 이번엔 실시간으로 동행을 구해 보겠습니다.

(맵 접속)

실시간에 들어오면, 근처에 있는 다른 유저들의 위치가 마커로 보입니다.

마침 바로 근처에 유저가 있네요.

(보성이 있음)

원하는 유저를 선택하면

(보성 클릭)

마찬가지로 프로필을 보고 채팅을 할 수 있습니다.

이후 과정은 게시글 동행과 동일하므로 생략하겠습니다.

(홈으로 이동)

(폰을 재완에게)

3일차 날이 밝았습니다.

게시글을 구경 중인데,

(본인인증 뱃지가 있는 유저의 게시글로 들어감)

이 분은 본인 인증 뱃지를 달고 있네요?

(페이지 이동)

본인 인증 페이지로 들어가서

얼굴 사진 3장을 찍습니다.

그리고 본인 인증을 누르면 잠시 기다린 후 본인 인증 뱃지가 발급됩니다.
인증된 유저가 됐으니, 동행과 매칭되기 더 수월하겠죠?

마지막으로 파리의 상징 에펠탑으로 떠나보겠습니다.

에펠탑에도 왔으니 낙서를 해볼까요?

당연히 진짜 낙서는 아니고 ar 낙서장을 활용해보겠습니다.

위치를 선택한 후, 낙서를 하고

(낙서: ^^)

드래그해서 원하는 위치에 낙서를 붙일 수 있습니다.

주변을 둘러보면 이전에 다른 유저가 남긴 낙서도 볼 수 있습니다.

(싸피 8기

모두 고생 많았어요!)

이상 시연 마치겠습니다. 감사합니다.