

목차

[1. 개발환경 및 외부서비스 2](#_Toc119427268)

[1.1. 프로젝트 개요 2](#_Toc119427269)

[1.2. 협업 툴 / 환경 2](#_Toc119427270)

[1.3. 개발환경 3](#_Toc119427271)

[1.4. 외부 서비스 3](#_Toc119427272)

[1.4.1. Amazon Web Service S3 (Simple Storage Service) 3](#_Toc119427273)

[1.4.2. Google Gmail SMTP 6](#_Toc119427274)

[1.4.3. Firebase Messaging Service 8](#_Toc119427275)

[2. 환경변수 설정 및 빌드 11](#_Toc119427276)

[2.1. 환경변수 설정 11](#_Toc119427277)

[2.1.1. Spring – application.properties 11](#_Toc119427278)

[2.1.2. Android 13](#_Toc119427279)

[2.2. 빌드 13](#_Toc119427280)

[2.2.1. Ubuntu Server- Backend 구축 13](#_Toc119427281)

[2.2.2. Android Application 설치 14](#_Toc119427282)

# 개발환경 및 외부서비스

## 프로젝트 개요

## 협업 툴 / 환경

* GitLab
  + Git-Flow. develop, FE\_develop, BE\_develop 브랜치를 나누고
  + 각 feature별 브랜치를 따서 작업 진행
  + MR시 Front/Back Maintainer가 확인 후 합병
* Notion
  + 회의가 있을때마다 회의록을 기록하여 보관
  + 컨벤션 정리
  + api 문서 관리 등
* JIRA
  + 매주 월요일 목표량을 설정하여 Sprint 진행
  + 업무별 Story Point(1~4)를 설정, In-Progress -> Done 순으로 작업
* MatterMost
  + Gitlab, JIRA 봇 연동하여 실시간으로 협업
  + Server 연동하여 실시간 에러 처리
* Webex
  + 회의 : 평일 아침 Webex에서 데일리 스크럼 진행
  + 문제점이 생겼을 때 팀원들에게 직접 소통

## 개발환경

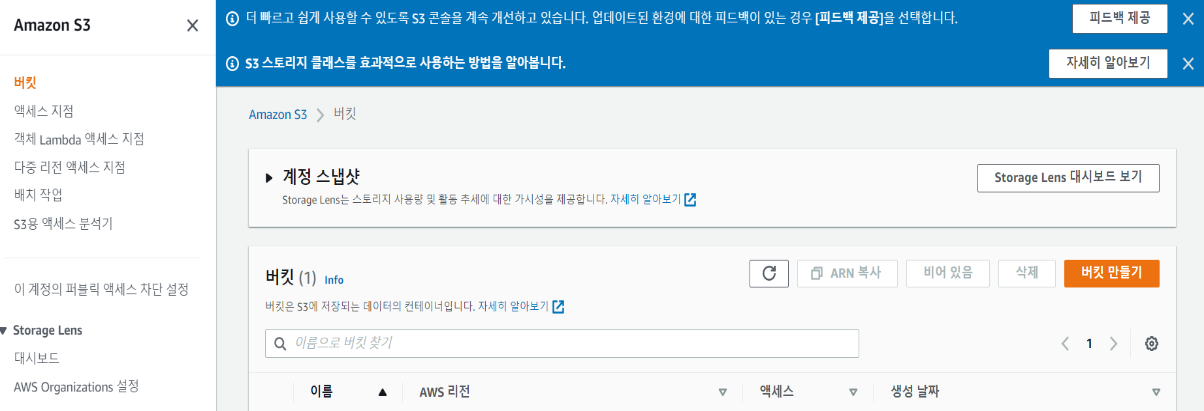
* **Backend - Spring**
  + IntelliJ IDEA : 2021.3.1
  + IntelliJ Runtime: 11.0.13+7-b1751.21 amd64
  + JDK : 17-ea
  + JRE : build 17-ea+14
  + JVM : build 17-ea+14, mixed mode, sharing
  + DB : 8.0.30-MySQL
  + Springboot : 2.7.3
  + Gradle : 7.5
* **Frontend – Android Studio**
  + Android Studio Dolphin | 2021.3.1 Patch 1
  + ART : 11.0.13+0-b1751.21-8125866 amd64
  + VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
  + SDK version : (min : 26, target : 32)
* **Unity :** 2021.3.9.f1
* **CI/CD**
  + Server : AWS EC2 Ubuntu 20.04 LTS
  + Docker : 20.10.20
  + nginx : 1.18.0
  + Jenkins : 2.346.2

## 외부 서비스

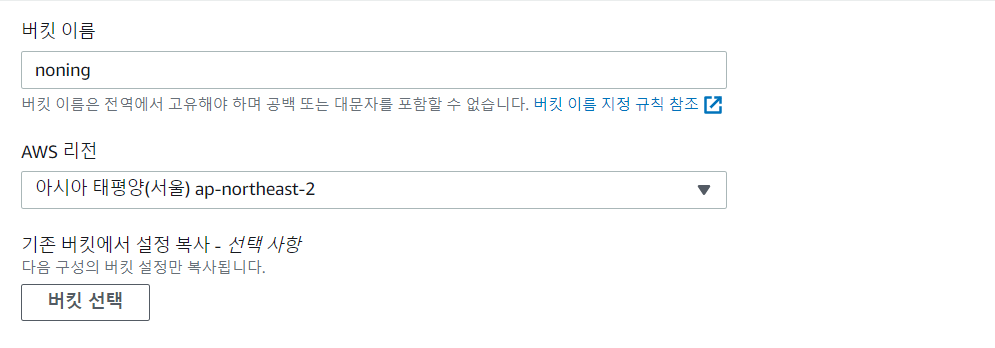
사용된 외부 서비스 : Amazon S3, Google Gmail SMTP, Firebase Service

### Amazon Web Service S3 (Simple Storage Service)

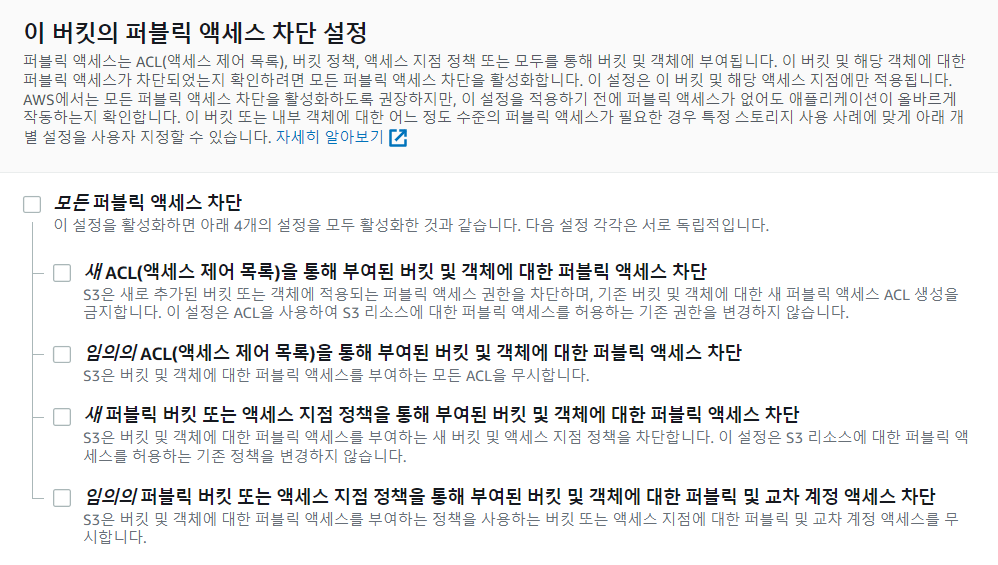
1. AWS 가입하기
2. S3 버킷 생성
   1. 버킷 만들기 선택



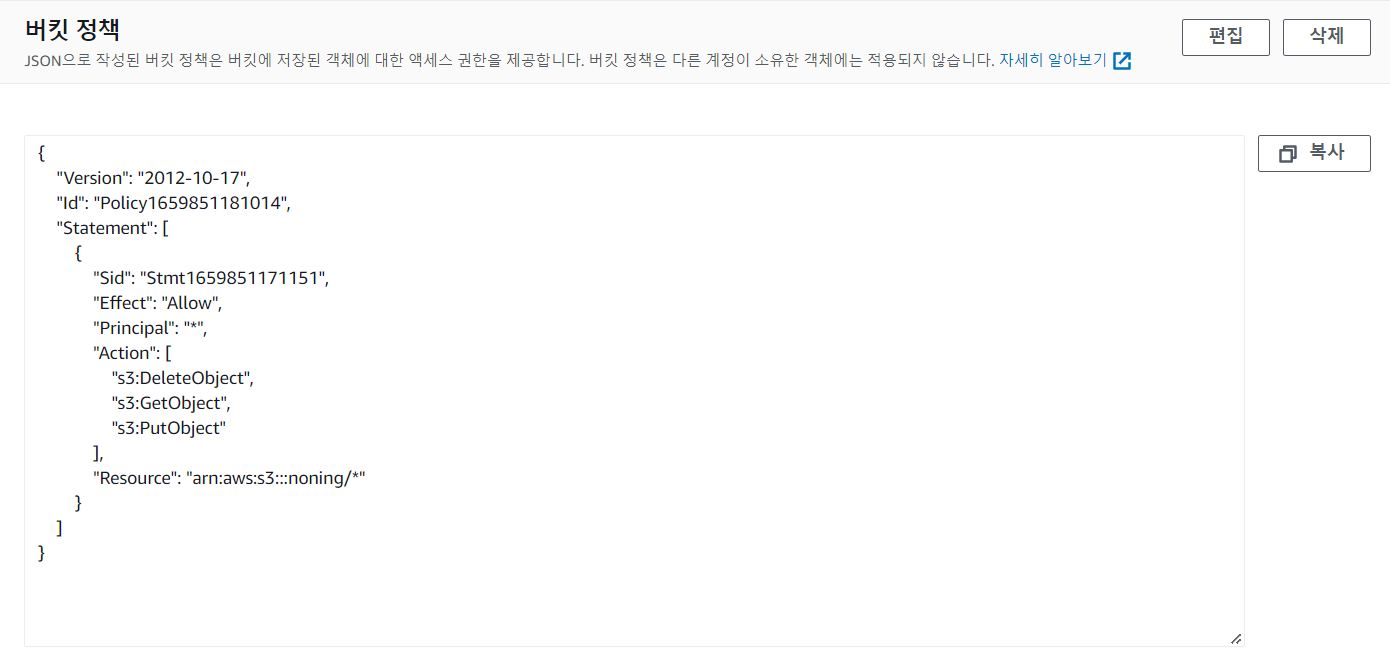
* 1. 버킷 이름 작성 및 AWS리전 서울 선택



* 1. 퍼블렉 엑세스 차단 풀어주기



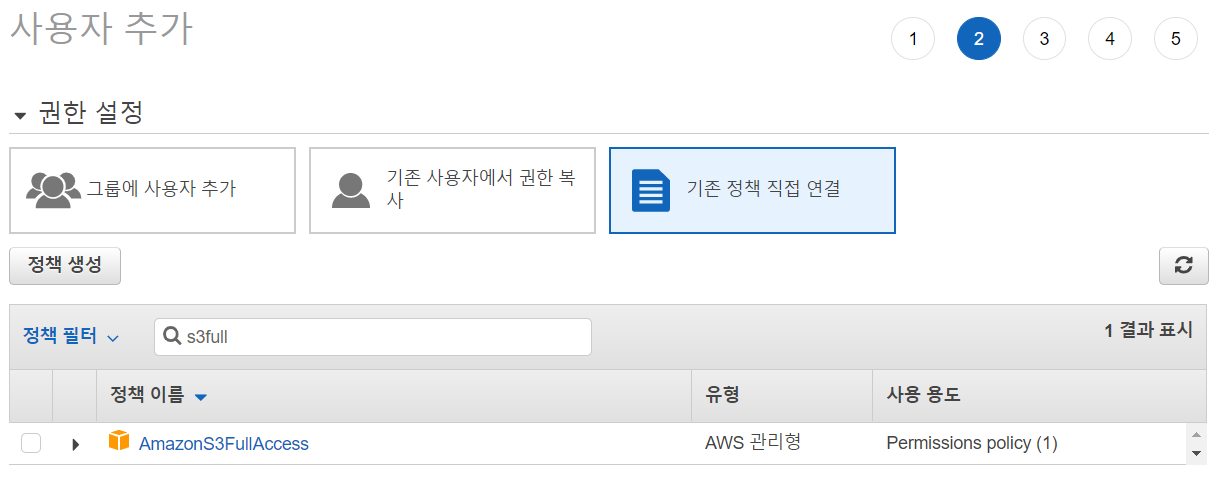
* 1. 버킷 만들기
  2. 버킷 – 권한에서 버킷 정책 설정



1. IAM 추가



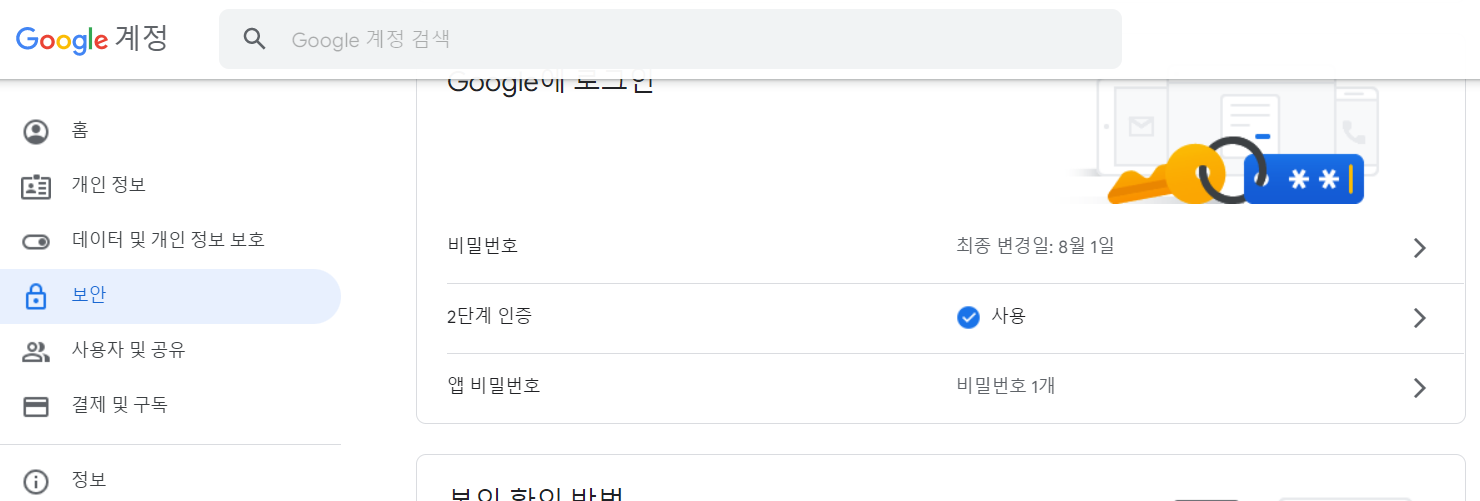
* 1. IAM 사용자 추가 – 엑세스 키 체크



* 1. S3FullAccess 정책 연결
  2. 생성된 사용자 csv파일 받은 후 Access Key, Secret key 확인

### Google Gmail SMTP

1. Google 메일 앱 비밀번호 발급
   1. Google 계정 설정 - 2단계 인증 설정



* 1. Google 계정 설정 – 보안 – 앱 비밀번호



* 1. 메일 , Windows 컴퓨터 비밀번호 생성
  2. Windows 컴퓨터용 앱 비밀번호 확인

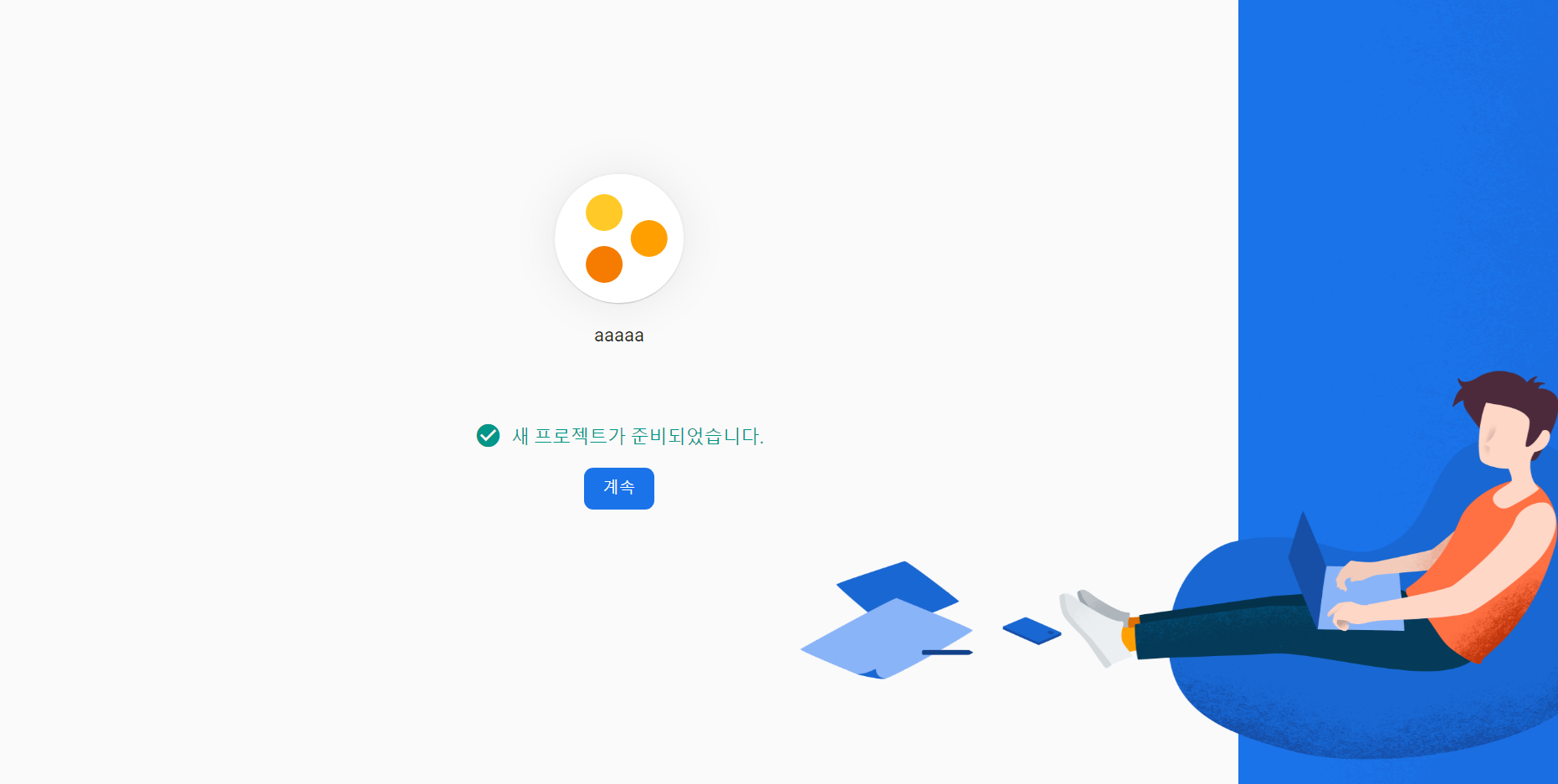
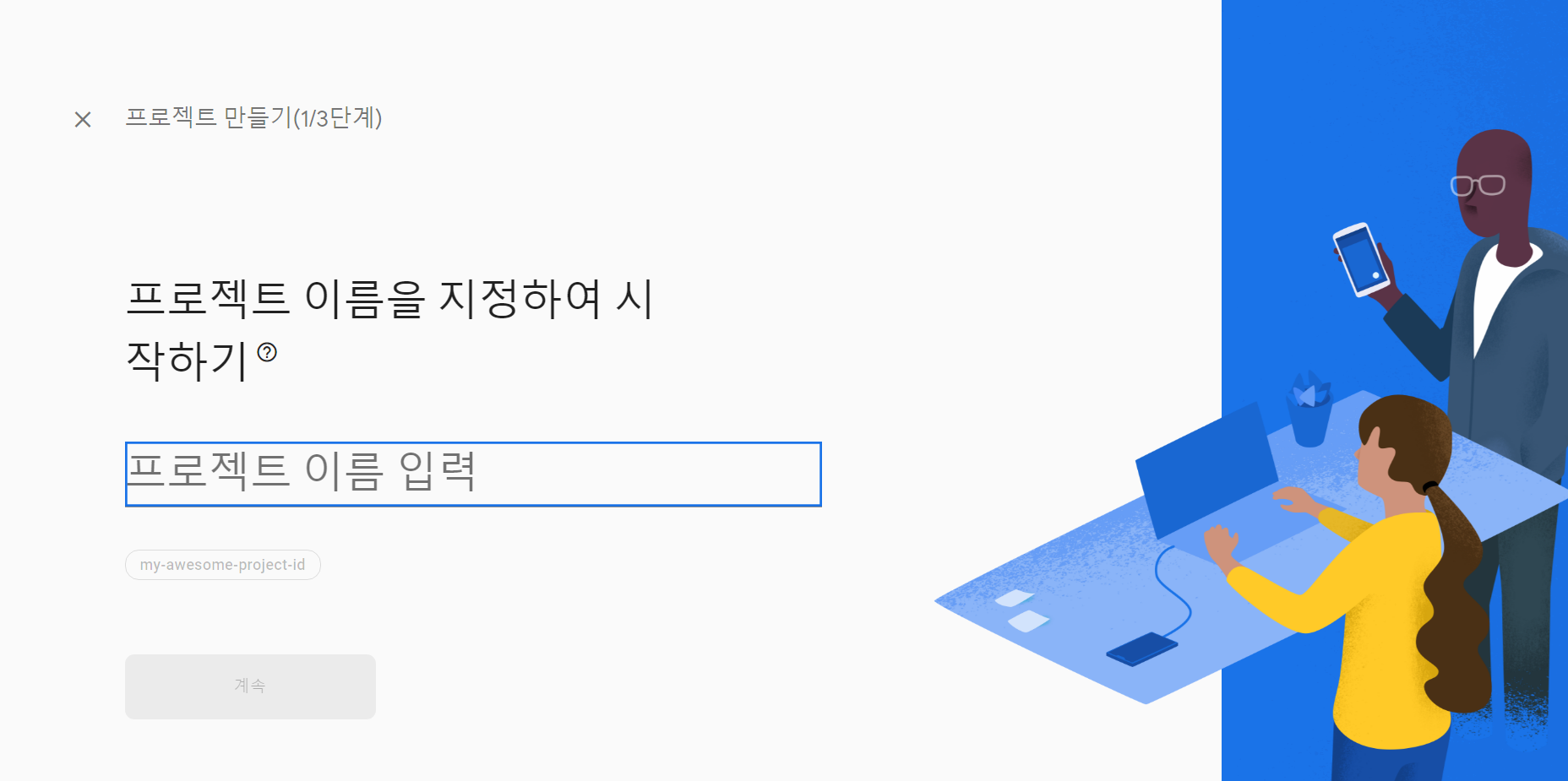
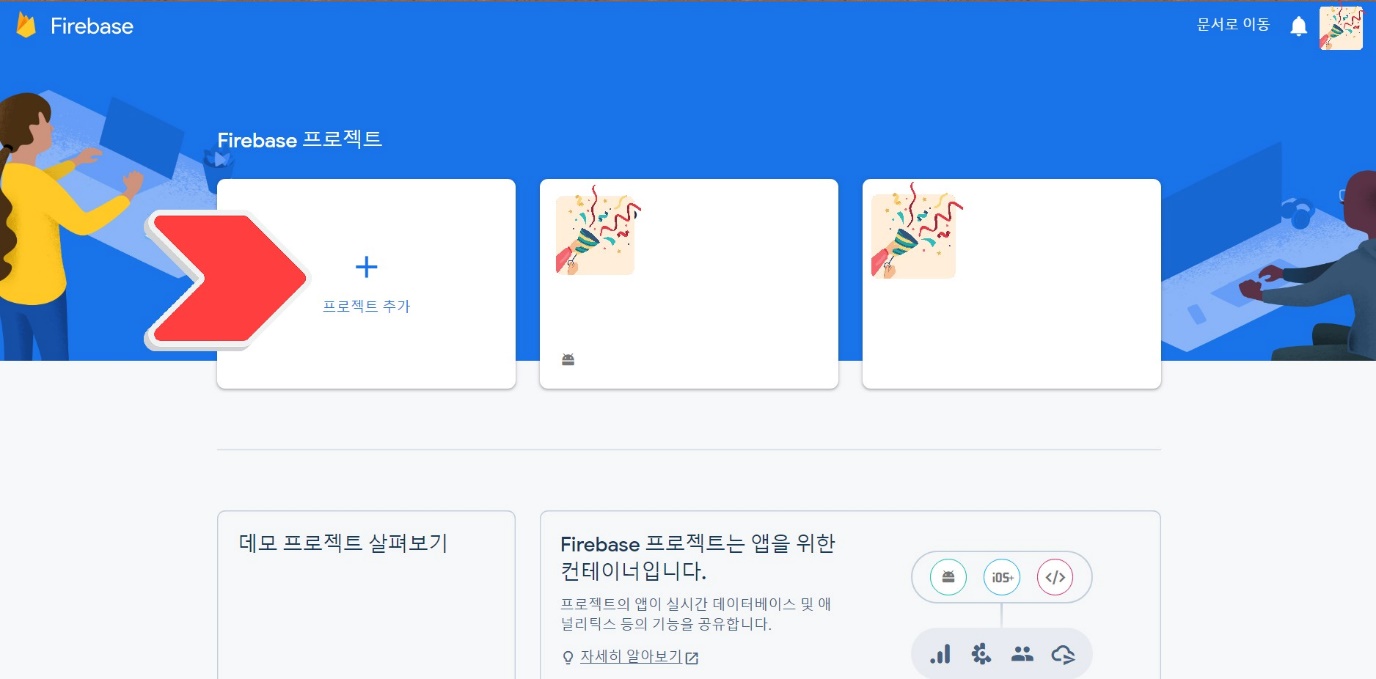
1. POP/IMAP 설정
   1. Gmail – 모든 설정



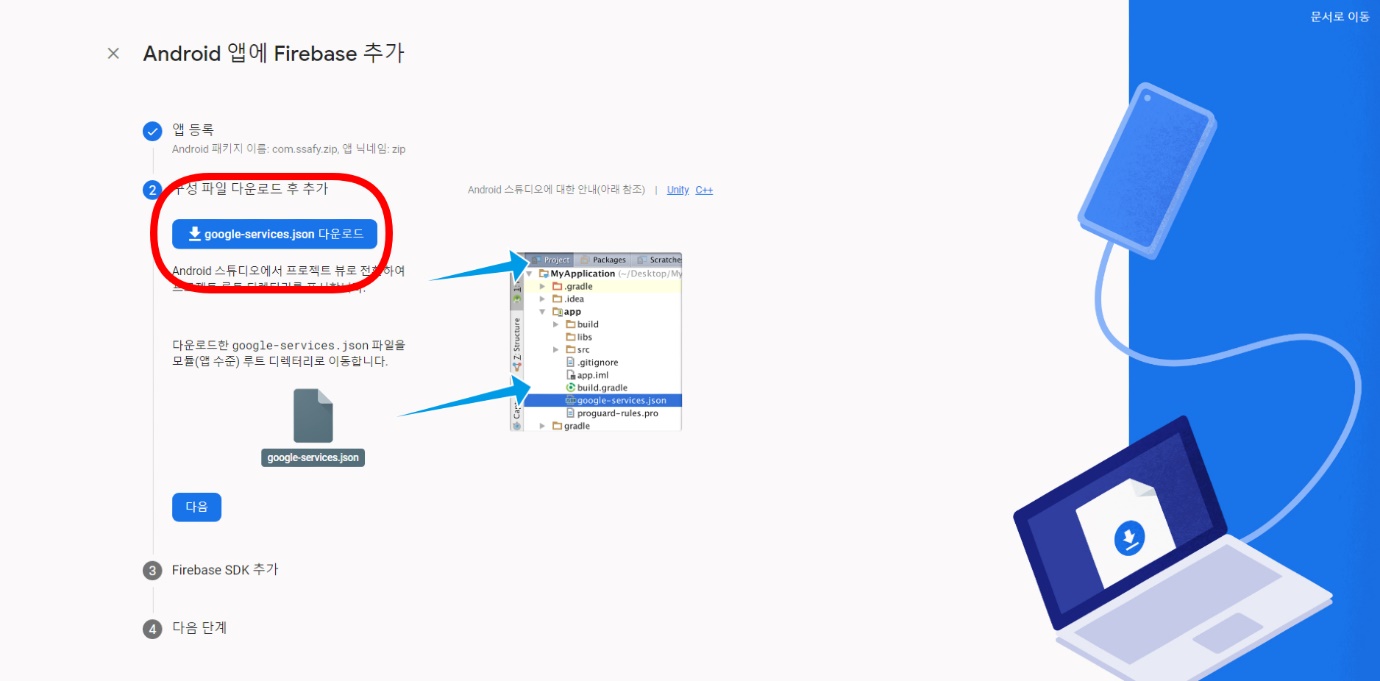
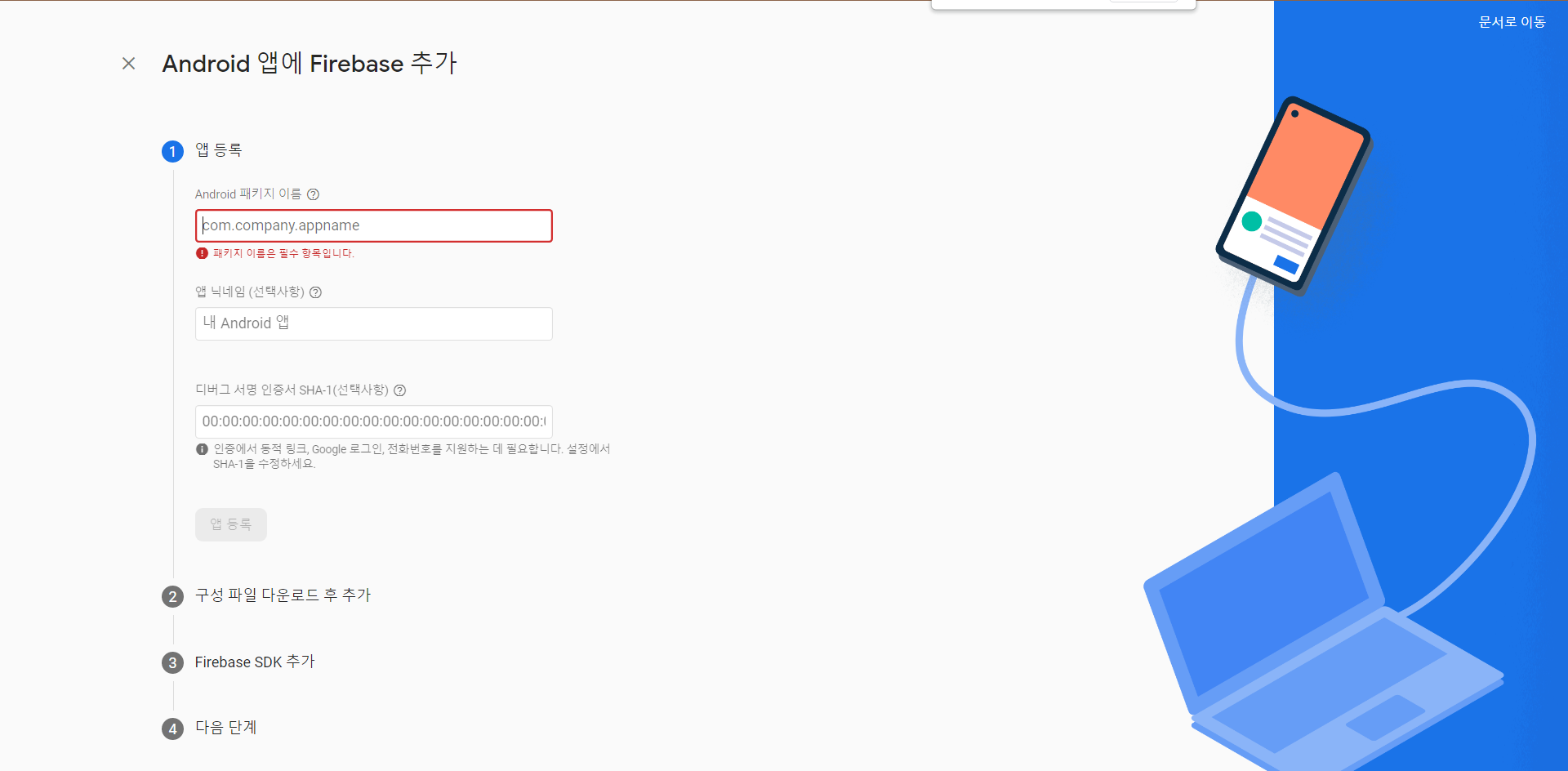
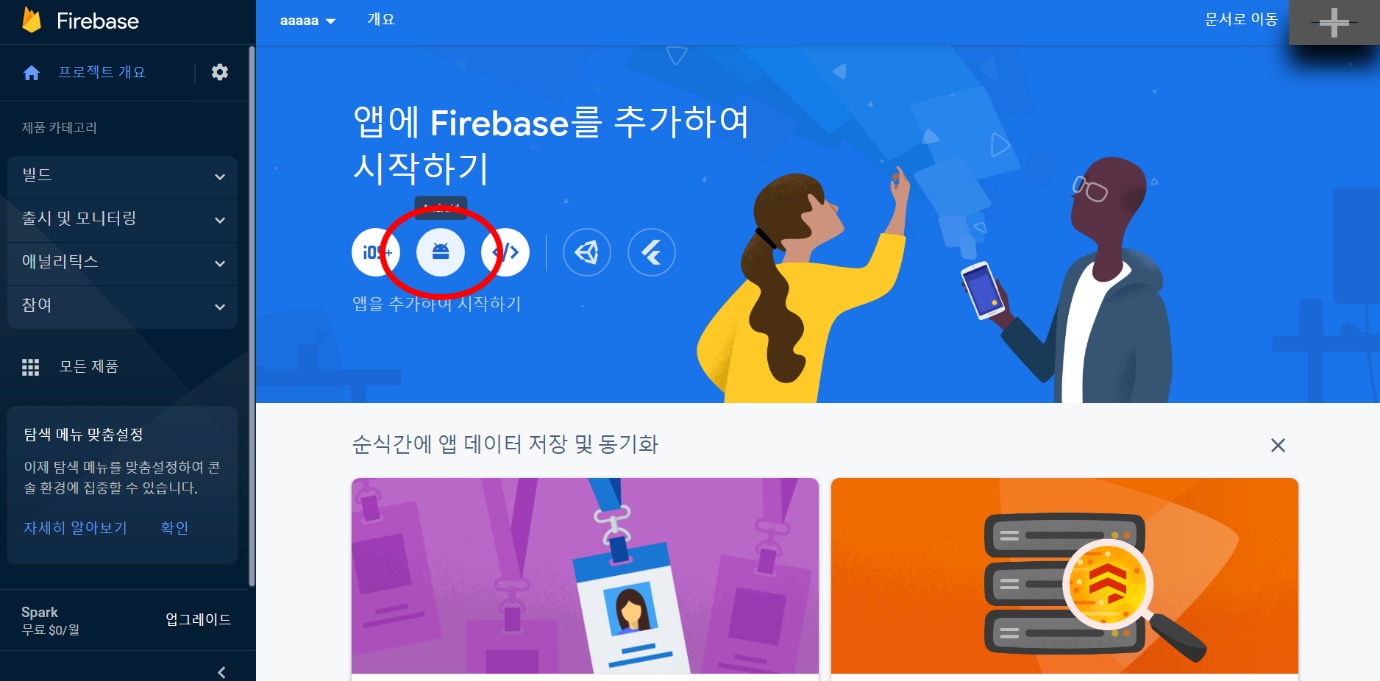
* 1. 전달 및 POP/IMAP 설정

### Firebase Messaging Service

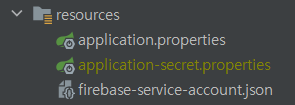
1. <https://console.firebase.google.com/>에서 프로젝트 추가



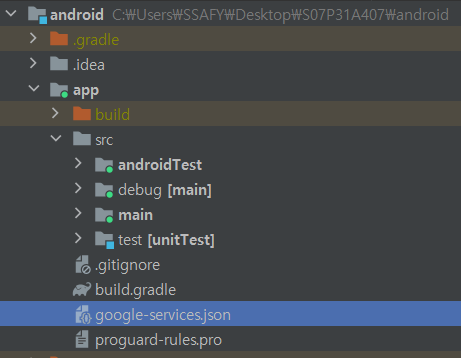
1. 프로젝트에 앱 추가 및 설정파일 다운로드



1. 다운로드 받은 google-service.json 파일 복사 및 붙여넣기
   * + - Spring resources 폴더



* + - * Andriod 프로젝트 내 위치



# 환경변수 설정 및 빌드

## 환경변수 설정

### Spring – application.properties

**#MySQL Settings**  
spring.datasource.url=jdbc:mysql://**${ DB source URL }**   
spring.datasource.username=**${ DB username }**   
spring.datasource.password=**${ DB password }**   
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQL5Dialect

**#Redis Settings**

spring.redis.host = 호스트 주소  
spring.redis.port = 포트 번호  
spring.redis.password= 레디스 암호

**# JWT**

JWT.SECRET : 시크릿 키

**#SMTP**

spring.mail.host : 주소 (구글 SMTP : smtp.gmail.com)

spring.mail.port : 포트 (구글 SMTP : 587)

spring.mail.username : 구글 ID

spring.mail.password : 메일 접속 비밀번호

spring.mail.properties.mail.smtp.auth : auth 명령 사용여부 ( true )

spring.mail.properties.mail.smtp.starttls.enable : TLS-protection 사용여부 ( true )

**#WEBHOOK**

notification.mattermost.enabled : 사용여부 ( true )

notification.mattermost.webhook-url : Webhook 주소

notification.mattermost.pretext : attachment의 상단에 나오는 텍스트

**#S3 Bucket**

cloud.aws.credentials.accessKey : 버켓 접근키

cloud.aws.credentials.secretKey : 버켓 비밀키

cloud.aws.stack.auto : 스택이름 자동 감지여부 ( false )

cloud.aws.s3.bucket : 버켓 이름

cloud.aws.region.static : 버켓 지역 (ap-northeast-2)

**#multipart 사이즈 설정**

spring.servlet.multipart.max-file-size=20MB

spring.servlet.multipart.max-request-size=20MB

**#기타 설정**

spring.mvc.pathmatch.matching-strategy=*ant\_path\_matcher*

spring.jpa.hibernate.ddl-auto=${스프링 설정에 맞는 DB-schema 관리, default=none, 테스트 용으로 자동 생성 위해 create}

### Android

**#Server URL 설정(com.ssafy.zip.android.ApiService)**

private const val BASE\_URL = "${Server URL}"

## 빌드

### Ubuntu Server- Backend 구축

// MySQL DB 구축

sudo apt-get update

sudo apt-get install mysql-server

// SQL 접속해 스키마 만들기

// Docker 설치

sudo apt-get update

sudo apt-get install **\**

ca-certificates **\**

curl **\**

gnupg **\**

lsb-release

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo **\**

"deb [arch=**$(**dpkg --print-architecture**)** signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

**$(**lsb\_release -cs**)** stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

apt-cache madison docker-ce | awk '5:20.10.20~3-0~ubuntu-jammy'

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin

// Docker에 레디스 컨테이너 구축

docker search redis

docker pull redis

docker run -p 6379:**${**output\_port\_num**}** --name redis -d redis:latest --requirepass "**${**password**}**"

// Docker에 Spring Server 컨테이너 구축

sudo apt-get install git

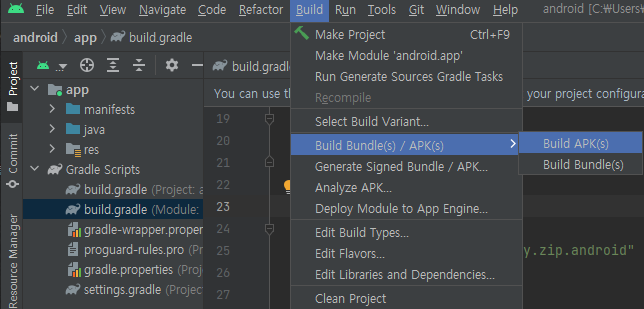
git clone https://lab.ssafy.com/s07-final/S07P31A407.git

cd ./backend

docker build -t zip:latest .

docker run --name zip\_back -d -p 8888:8888 zip:latest

### Android Application 설치

\android\app\build\outputs\apk\release 폴더에 생성 -> 안드로이드 폰 다운로드 후 실행