시스템 분석 설계 포트폴리오

컴퓨터정보공학과 20190746 박재윤

목차 A table of contents

Part 1,

파이어 베이스 연결

1. 파이어 베이스 연결

파이어 베이스 소개

파이어 베이스란?

파이어베이스(Firebase)는 2011년 파이어베이스(Firebase, Inc) 사가 개발하고 2014년 구글에 인수된 모바일 및 웹 애플리케이션 개발 플랫폼이다.

선택한 이유

1. 서버 인프라를 따로 구축할 필요가 없다.
- Realtime Database(실시간 데이터베이스), Authentication(인증), Storage(저장소),Hosting(호스팅), Firestore(신버전 데이터베이스) 기능으로 이미 구축 되어있다.

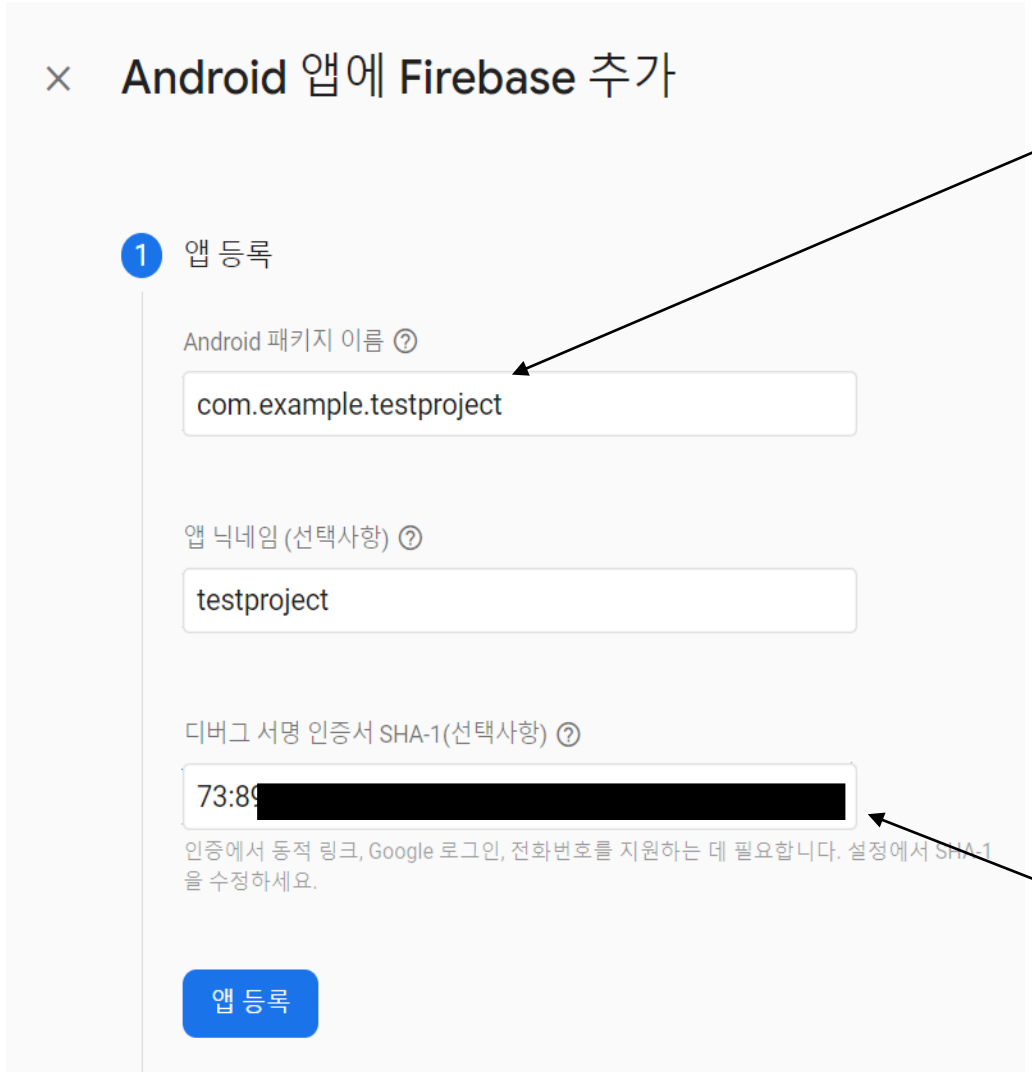
2.무료이용이 가능하다.
-특정 사용량까지는 무료로 이용 가능하며 사용량에 따라 종량제로 유로로 사용이 가능하다.

3.사용방법이 쉽다.
- 백엔드 개발 지식이 없어도 쉽게 개발이 가능하다.

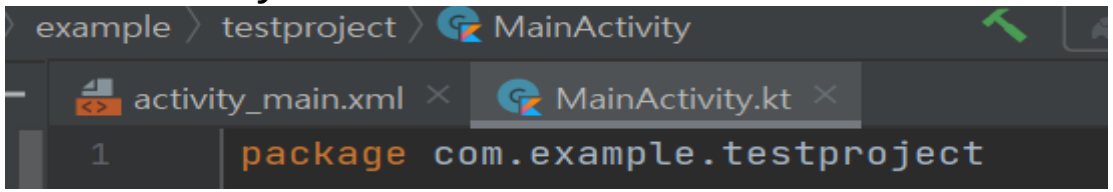
1. 파이어 베이스 연결

파이어 베이스 연결 과정

안드로이드 스튜디오와 파이어 베이스 연결



MainActivity에서 패캐지명을 그대로 적음

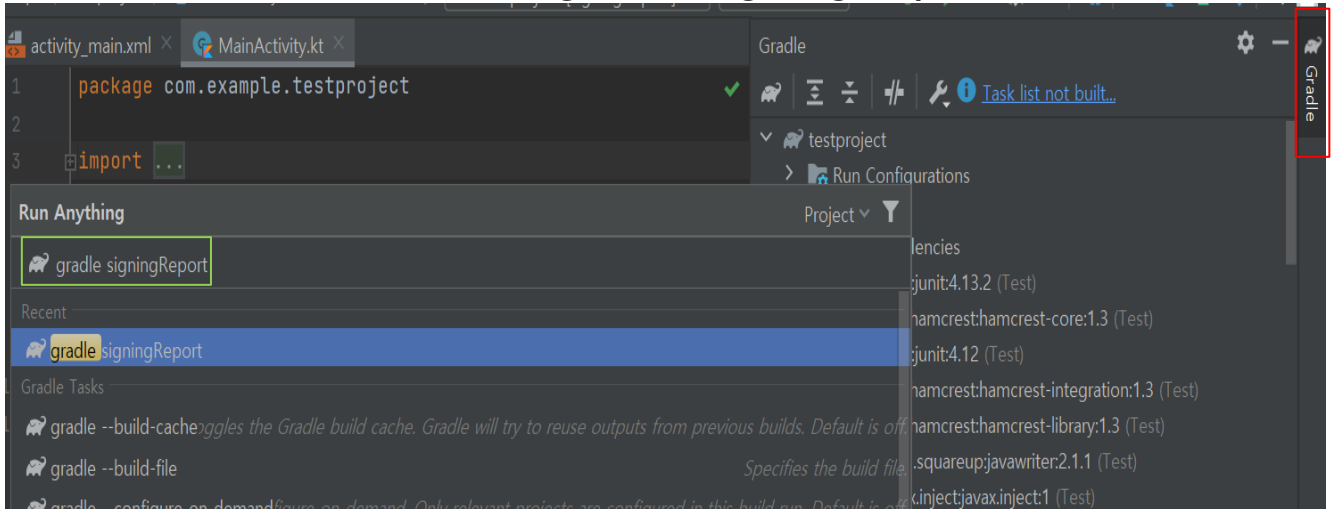


디버그 서명 인증서 SHA-1 얻는법

1. 오른쪽 Gradle클릭
2. 코끼리 모양 클릭하여 gradle signingReport 입력

1

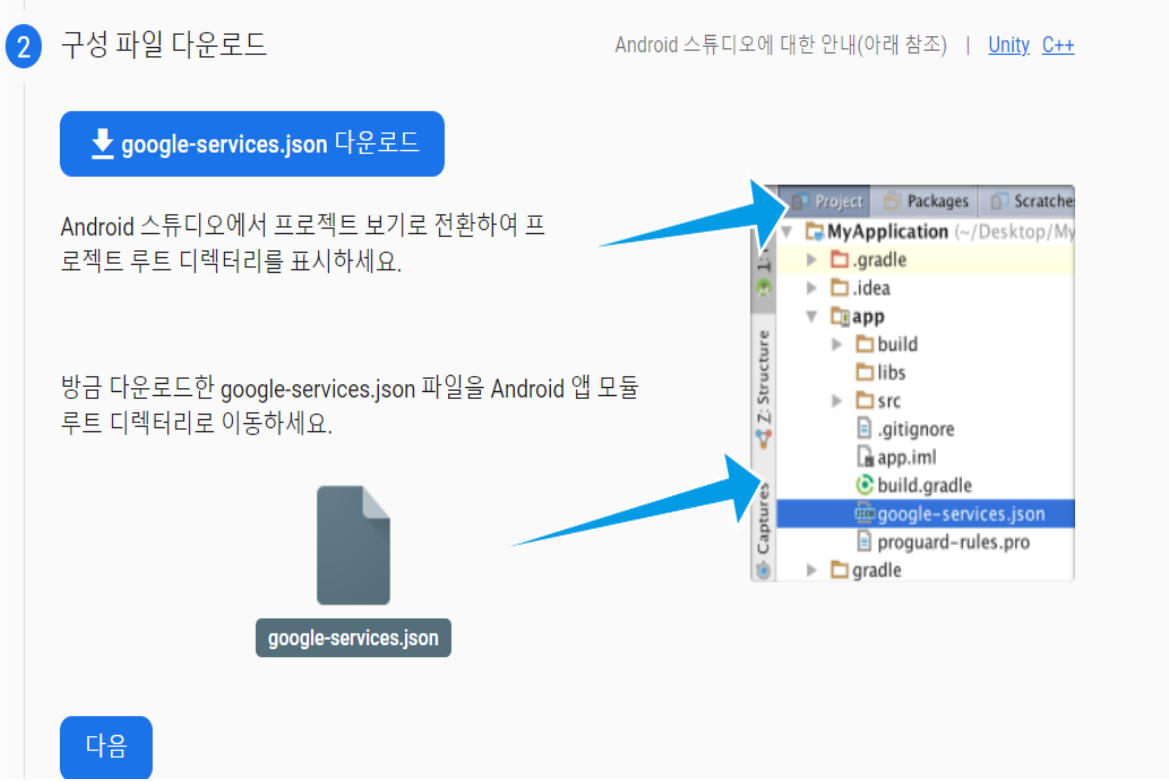
2



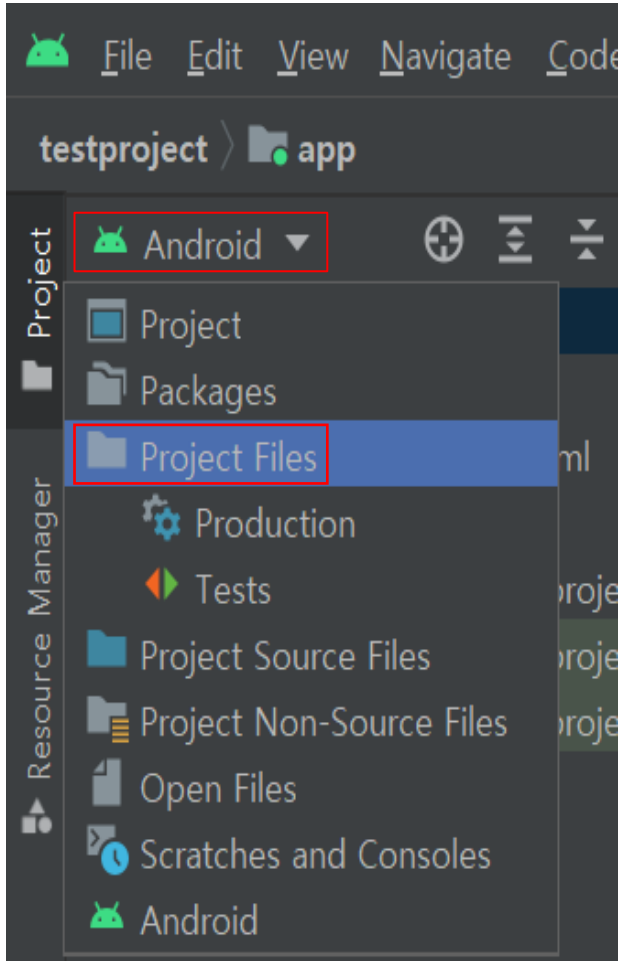
1. 파이어 베이스 연결

파이어 베이스 연결 과정

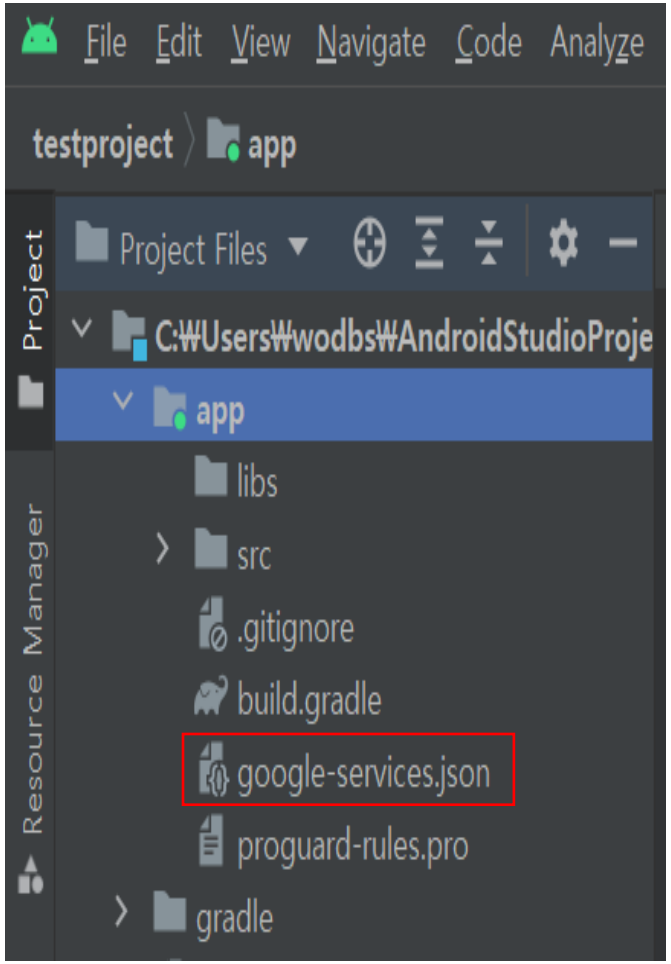
구성 파일 다운 받아 프로젝트 파일로 이동



파일을 보기 편하게
Project Files로 이동



다운받은 파일을 app아래에 저장



1. 파이어 베이스 연결

파이어 베이스 연결 과정

Firebase SDK 추가 하기위해 Gradle파일을 수정

3 **Firebase SDK 추가** Gradle 안내 | Unity C++

Gradle 용 Google 서비스 플러그인에서 방금 다운로드한 google-services.json 파일을 로드할 수 있습니다. 플러그인을 사용하려면 build.gradle 파일을 수정하세요.

프로젝트 수준의 **build.gradle** (<project>/build.gradle):

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.10'
  }
}

allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

Arctic Fox 버전 이후 추가 안함

※안드로이드 스튜디오 버전에 따라 방법이 조금씩 다름

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath "com.android.tools.build:gradle:7.0.4"
        classpath 'com.google.gms:google-services:4.3.10'//코드추가
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

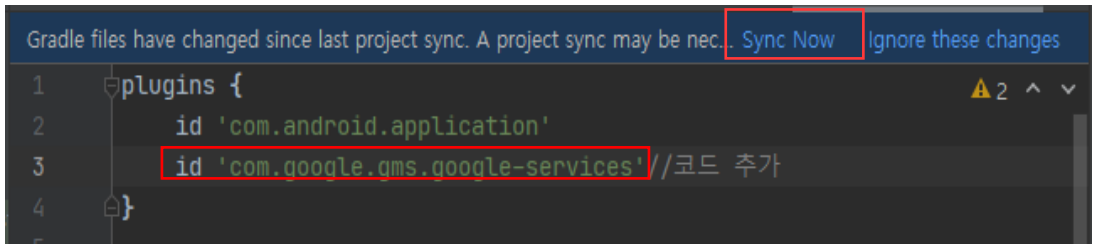
1. 파이어 베이스 연결

파이어 베이스 연결 과정

Firebase SDK 추가 하기위해 Gradle파일을 수정


```
plugins {
    id 'com.android.application'
    id 'com.google.gms.google-services'//코드 추가
}

android {
    compileSdk 31

    defaultConfig {
        applicationId "com.example.testproject"
        minSdk 26
        targetSdk 31
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.6.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation 'com.google.firebase:firebase-database:19.2.1'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation platform('com.google.firebase:firebase-bom:30.0.2')//코드추가
    implementation 'com.google.firebase:firebase-analytics'//코드추가

}
```

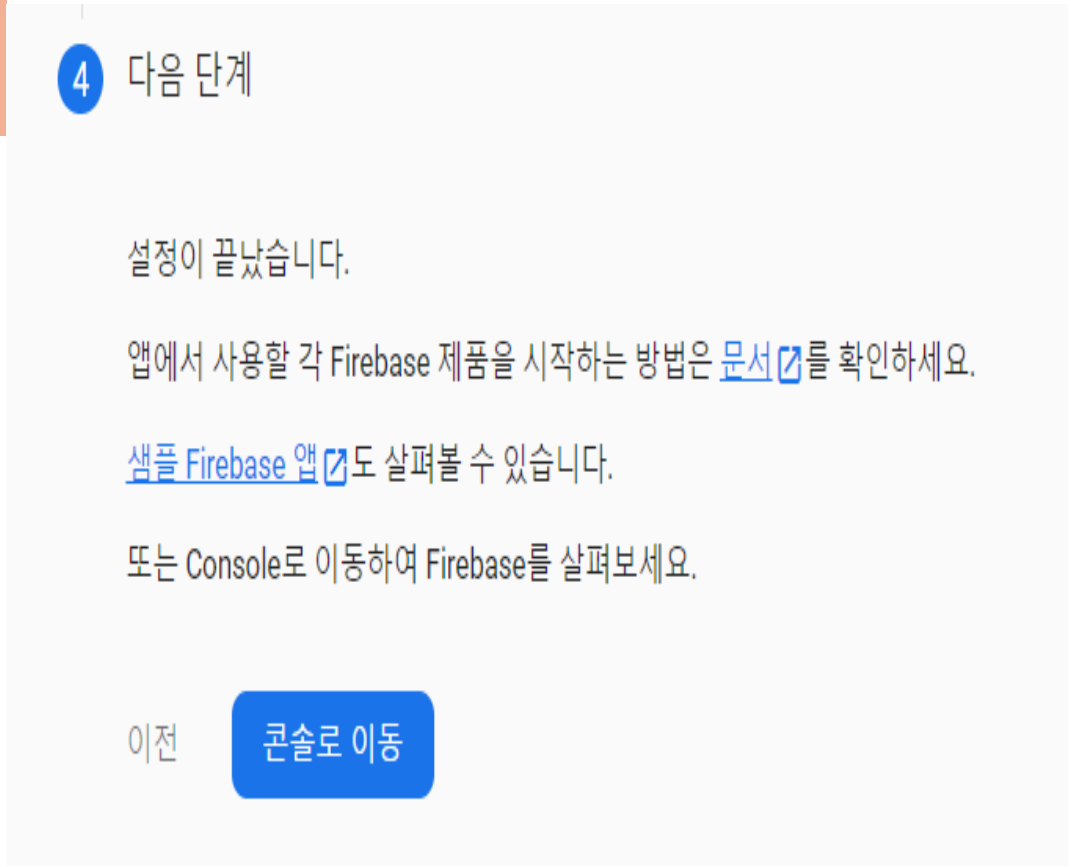
코드 추가 후 Sync Now를 클릭하여 프로젝트를 Gradle파일과 동기화



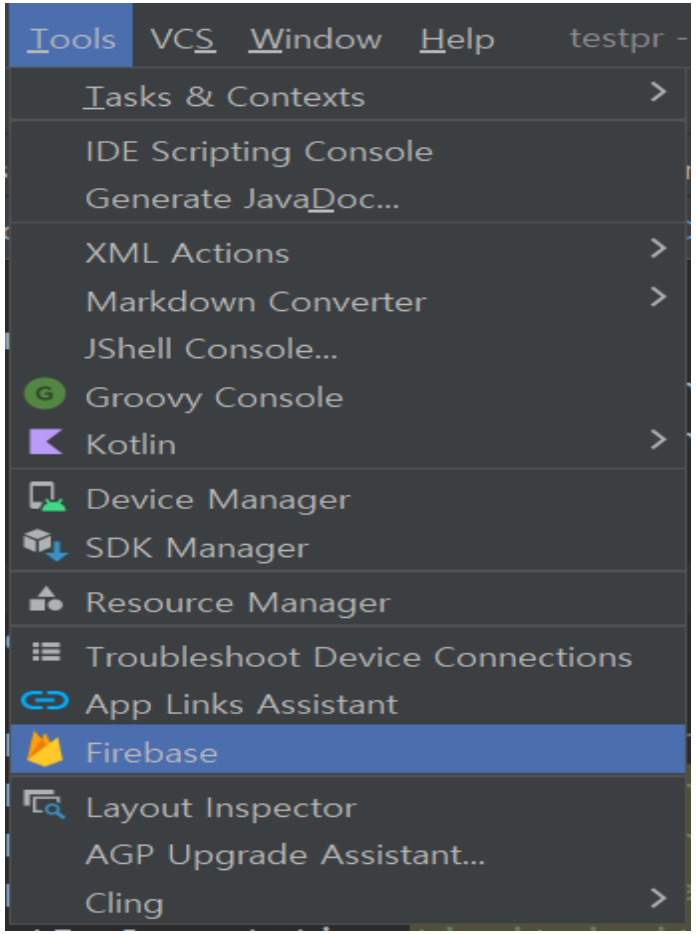
1. 파이어 베이스 연결

파이어 베이스 연결 과정

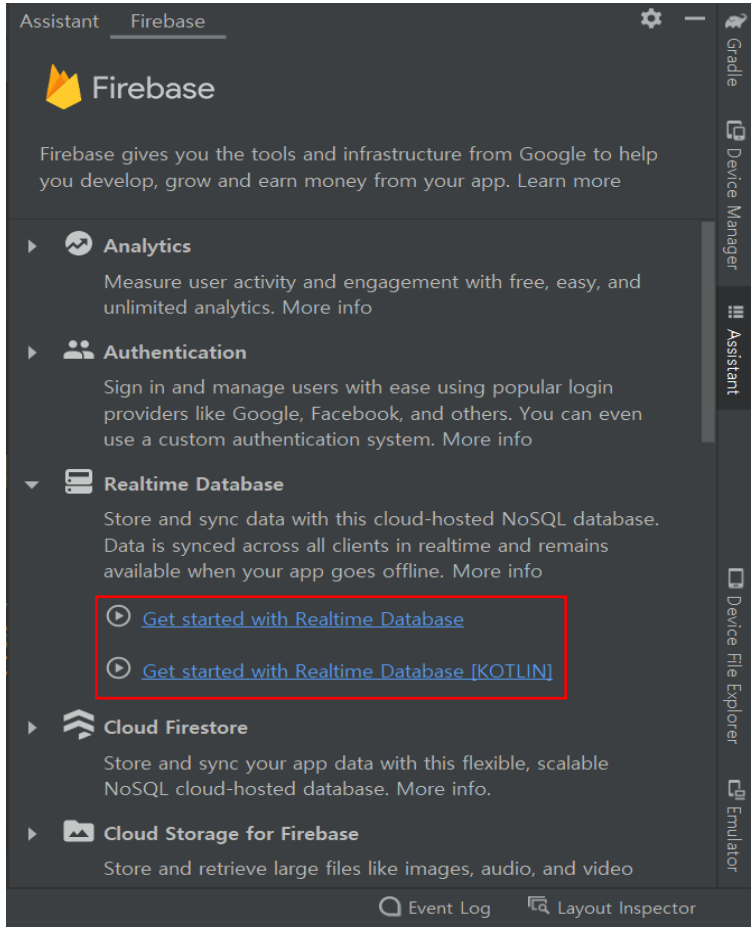
설정완료후 콘솔로 이동 클릭



안드로이드와 파이어 베이스가 제대로 연결되었는지 확인



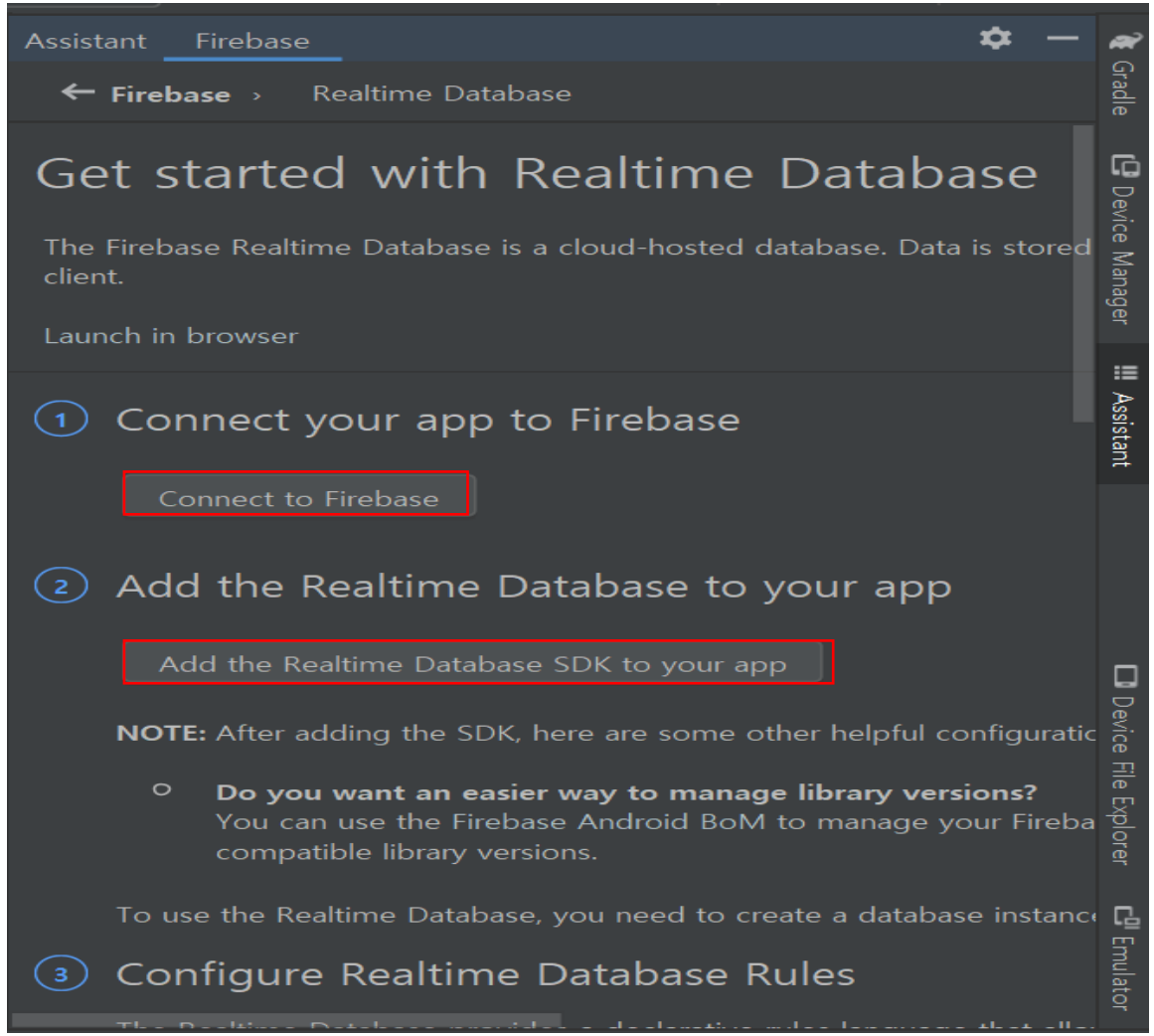
상단의 Tools에서 Firebase를 선택



사용할 기능선택

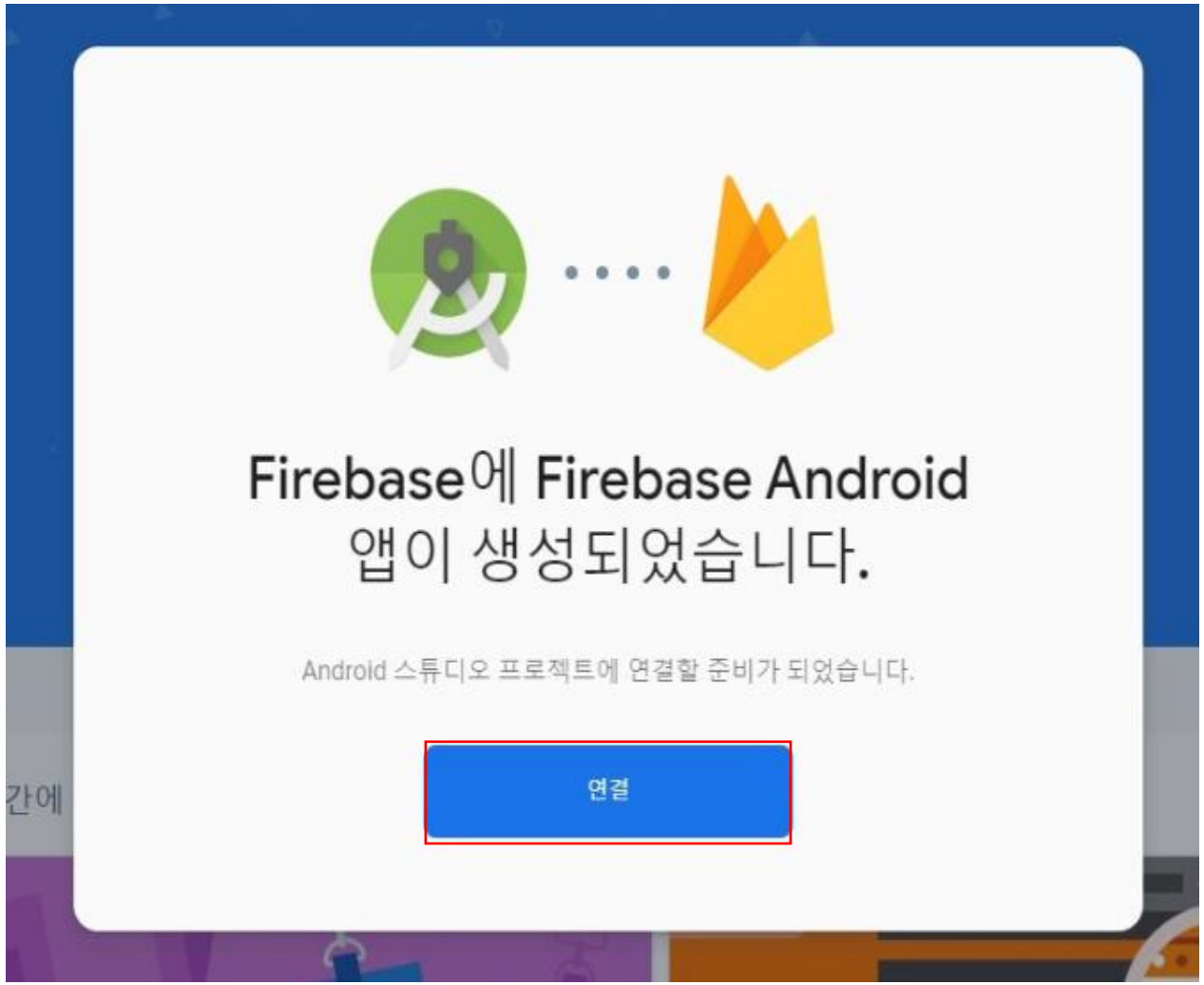
1. 파이어 베이스 연결

파이어 베이스 연결 과정



순차적으로 진행하여1,2번 모두 connected상태로 전환

오류 없이 제대로 수행 시 연결 성공

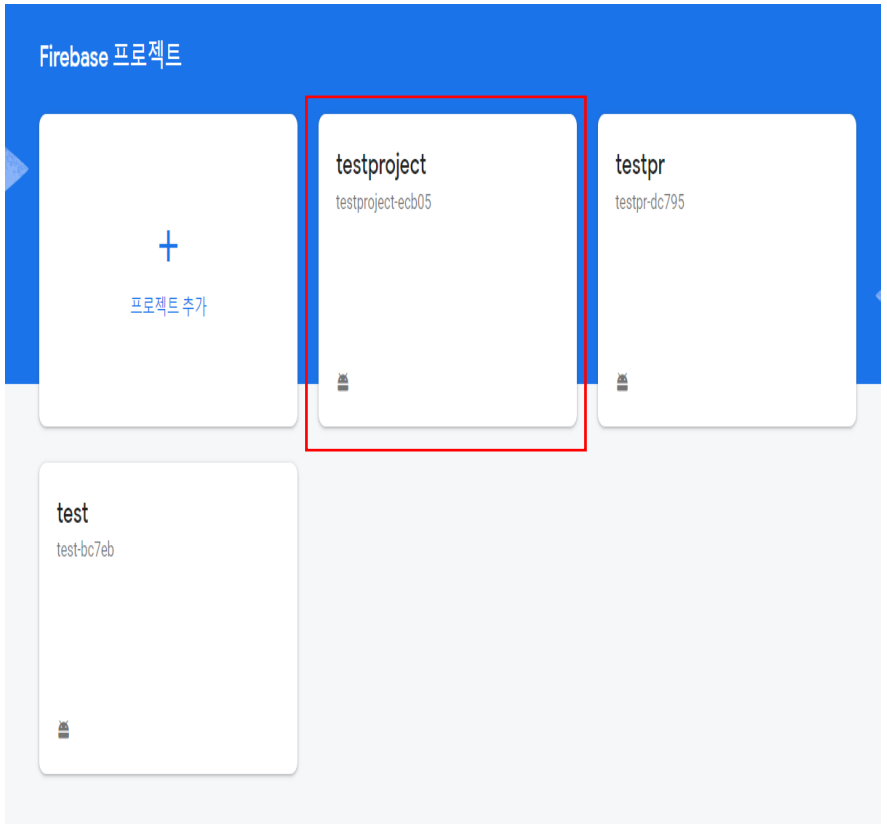


Part 2,

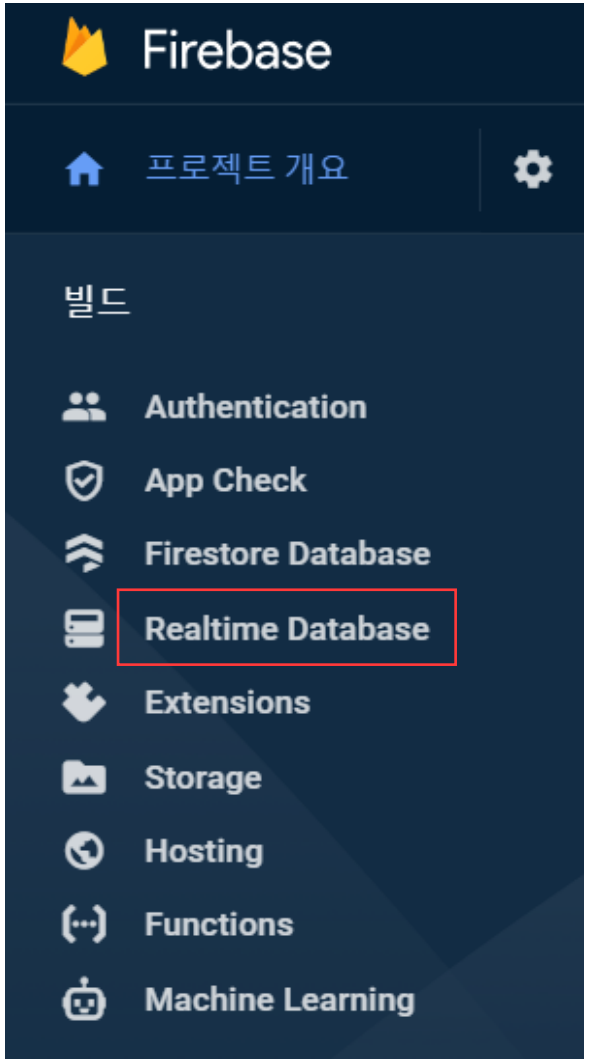
파이어 베이스 연동

2. 파이어 베이스 연동

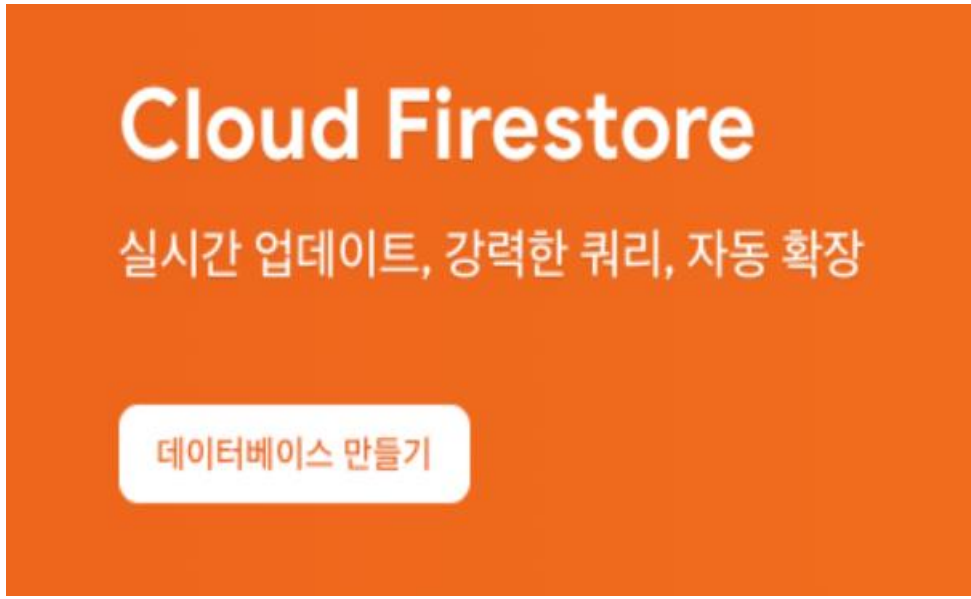
연결된 프로젝트 선택



Realtime Database선택



데이터베이스 생성



2. 파이어 베이스 연동

사용자가 권한 없이 데이터를 읽고 쓰기를 하기 위해서 Firebase의 Realtime Database의 규칙을 설정

인터넷 권한 허용

파이어 베이스에 접근하기 위해 안드로이드 스튜디오의 AndroidManifest에서 인터넷 권한을 허용

```
<!-- 인터넷 사용 권한 -->
<uses-permission android:name="android.permission.INTERNET" />
```

2. 파이어 베이스 연동

데이터를 추가하거나 조회할 때 사용하는 데이터 클래스 생성

@IgnoreExtraProperties는 추가 속성 무시로 맵핑할 때 번지수가 맞는 것 만을 넣어준다는 의미

값을 추가할 때 사용할 함수를 선언

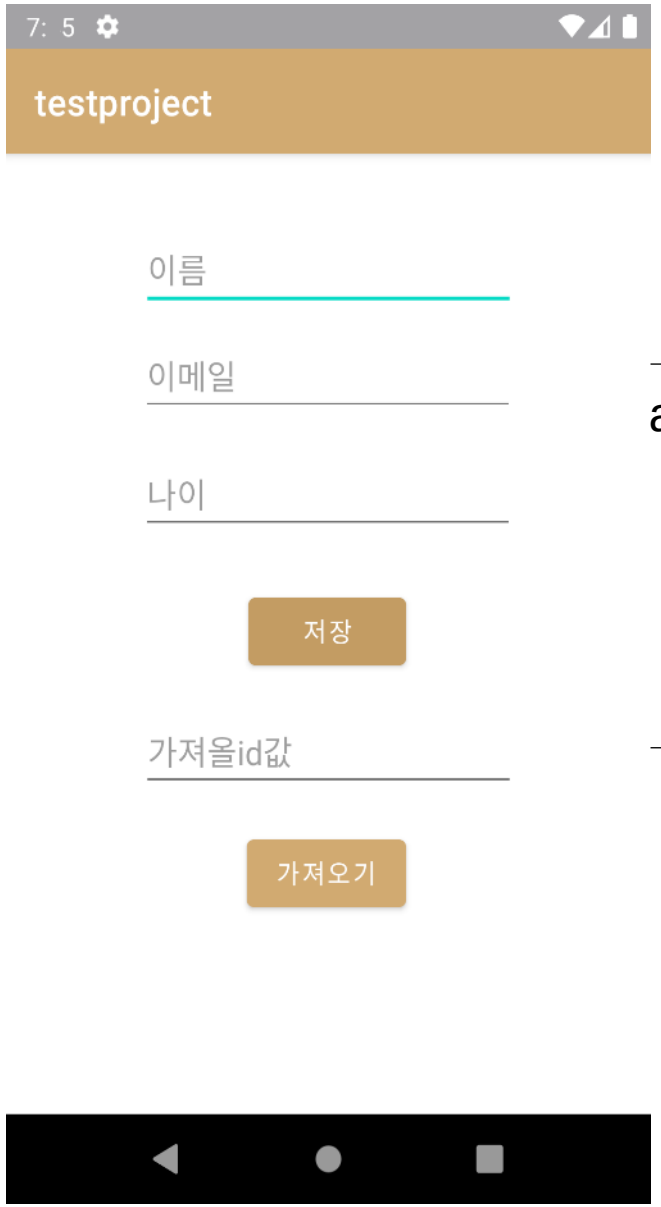
생성자 선언

Getter, Setter 설정

```java
package com.example.testproject;

import com.google.firebase.database.IgnoreExtraProperties;

@IgnoreExtraProperties//안드로이드와 파이어베이스를 맵핑할 때 번지수가 맞는거만 넣어준다는 의미의 어노테이션
public class User {
    public String name; //이름
    public String email; //이메일
    public String age; //나이

    public User(){} //생성자

    //값을 추가 할때 쓰는 함수 MainActivity 에서 함수로 사용
    public User(String name, String email, String age) {
        this.name = name;
        this.email = email;
        this.age = age;
    }
    //getter, setter 설정
    public void setName(String name) {
        this.name = name;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public void setAge(String age) {
        this.age = age;
    }
    public String getName() {
        return name;
    }
    public String getEmail() {
        return email;
    }
    public String getAge() {
        return age;
    }
}
```

2. 파이어 베이스 연동

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <EditText
        android:id="@+id/editName"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="이름"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.085" />

    <EditText
        android:id="@+id/editAge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="나이"
        android:inputType="number"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/editName"
        app:layout_constraintVertical_bias="0.195" />

    <Button
        android:id="@+id/buttonSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:backgroundTint="#C39C63"
        android:text="저장"
```



activity_main.xml에서 화면 구성

EditText, Button, TextView를 이용하여 화면 구성

2. 파이어 베이스 연동

MainActivity 설정

파이어 베이스와 연결하여 데이터를 읽거나 쓰기 위해 DatabaseReference가 필요

사용할 변수들을 선언하고 각각의 id값을 읽어와 변수에 연결

Int i를 이용하여 각각의 데이터를 구별하기 위해 PK(Primary Key)로 사용

```java
package com.example.testproject;

import ...


public class MainActivity extends AppCompatActivity {

    private DatabaseReference mDatabase;
    //파이어베이스에 데이터를 추가나 조회하기 위해 사용

    Button save, read;
    EditText email, name, age, id;
    TextView data;
    int i = 1; //primary Key

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mDatabase = FirebaseDatabase.getInstance().getReference();
        //데이터를 읽거나 쓰기위해서는 DatabaseReference의 인스턴스가 필요
        save = findViewById(R.id.buttonSave); //저장 버튼
        read = findViewById(R.id.buttonRead); //가져오기 버튼
        name = findViewById(R.id.editName); //이름
        email = findViewById(R.id.editEmail); // 이메일
        age = findViewById(R.id.editAge); // 나이
        id = findViewById(R.id.editID);// id
        data = findViewById(R.id.textData); // 가져올 데이터
```

2. 파이어 베이스 연동

```
private void writeUser(String userId, String name, String email, String age){ //함수
    User user = new User(name, email, age);
    //데이터 저장
    mDatabase.child("users").child(userId).setValue(user)
            .addOnSuccessListener(new OnSuccessListener<Void>() { //데이터베이스에 넘어간 이후 처리
                @Override
                public void onSuccess(Void aVoid) {
                    Toast.makeText(getApplicationContext(),"저장을 완료했습니다", Toast.LENGTH_LONG).show();
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(getApplicationContext(),"저장에 실패했습니다" , Toast.LENGTH_LONG).show();
                }
            });
}
```

데이터 저장 함수 선언

클래스 파일에서 만들었던 함수사용

child는 해당 키 위치로 이동하는 함수이고 키가 없으면 매개 변수 값을 이용하여 키를 생성

setValue는 지정된 주소에 데이터를 저장

addOnSuccessListener는 데이터가 넘어간 이후 처리를 담당(성공,실패)

Toast는 잠깐 동안 지정된 값을 보여줌

2. 파이어 베이스 연동

데이터 읽기 함수선언

addValueEventListener메소드는 데이터베이스 내부 값이 변경될 때마다 호출

onDataChange메서드는 이벤트 발생 시점에 특정 경로에 있던 컨텐츠의 정적 스냅샷을 읽음

스냅샷이란? 사진처럼 특정 시점에 스토리지의 파일 시스템을 포착해 보관하는 기술을 의미

스냅샷에 getValue를 호출하면 스냅샷이 가지고 있는 데이터를 얻어 옴

읽어온 데이터를 setText를 이용하여 표현

onCancelled는 읽기가 취소되면 호출

```
    private void readUser(String userId) {//함수
        //데이터 읽기
        mDatabase.child("users").child(userId).addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                User user = snapshot.getValue(User.class);

                data.setText("이름: " + user.name + "\n이메일: " + user.email + "\n나이: " + user.age);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }
}
```

2. 파이어 베이스 연동

```
save.setOnClickListener(new View.OnClickListener() {// 저장 버튼누르면 값저장
    @Override
    public void onClick(View v) {
        String getUserName = name.getText().toString();
        String getUserEmail = email.getText().toString();
        String getUserAge = age.getText().toString();
        writeUser(Integer.toString(i++), getUserName, getUserEmail, getUserAge);//함수
    }
});

read.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        readUser(id.getText().toString());

    }
});
}
```

이벤트 처리

저장 버튼을 누르면 데이터들을 가져와 문자열로 바꾸고 선언했던 함수를 호출하여 매개변수로 값을 넘김 (앞에서 i = 1로 선언, 저장 버튼이 눌릴 때마다 첫번째 매개변수를 1씩 증가 시켜 값을 넘겨서 PK로써 데이터를 구별)

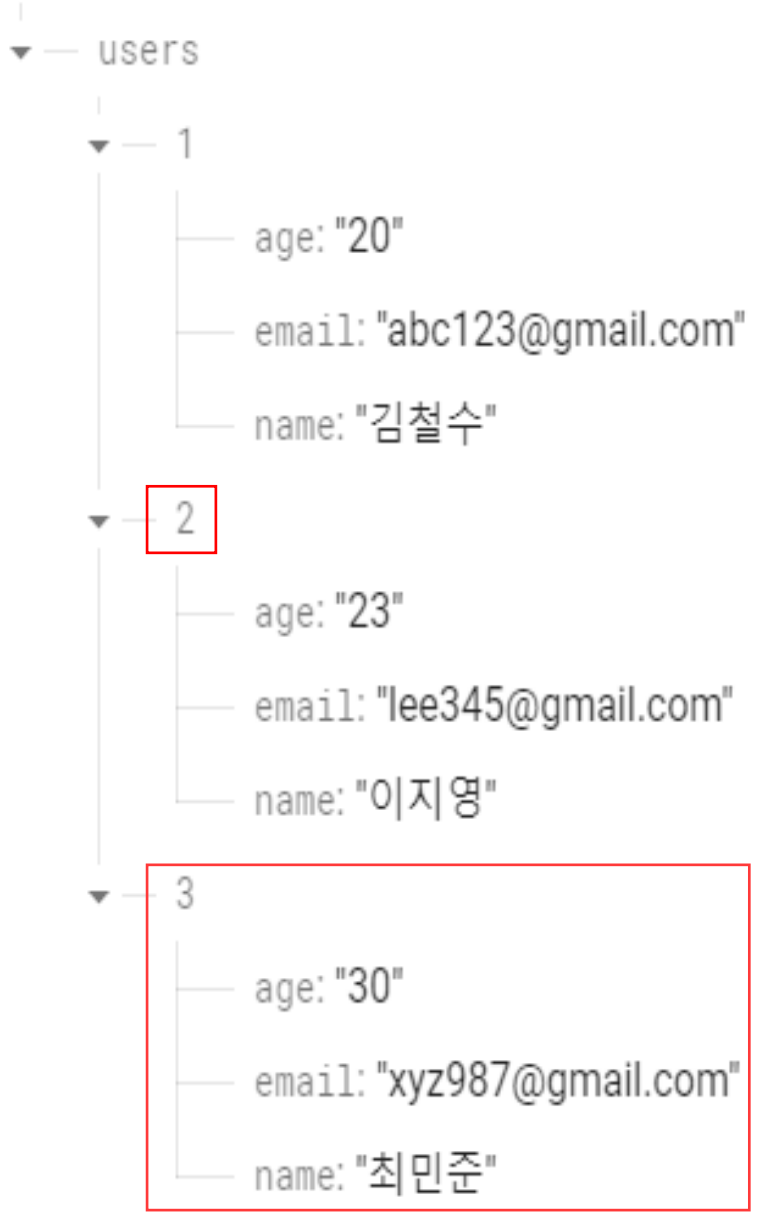
가져오기 버튼을 누르면 아이디 값을 문자열로 바꾸어 함수의 매개변수로 넘겨 호출

2. 파이어 베이스 연동

동작

데이터를 입력하고 저장 버튼을 누르면 파이어 베이스에 실시간으로 데이터가 추가

가져올 데이터의 아이디 값을 입력한 후 가져오기 버튼을 누르면 아래에 id값에 맞는 데이터를 형식에 알맞게 가져옴



Part 3,

JSON과 파이어 베이스

3. JSON과 파이어 베이스

NoSQL에 대하여

NoSQL이란?

비관계형 데이터 베이스
관계형 데이터 모델을 **지양** 하며 대량의 분산된 데이터를 저장하고 조회하는 데 특화되었으며 스키마 없이 사용 가능하거나 느슨한 스키마를 제공하는 저장소를 의미

스키마란? 데이터베이스의 구조와 제약 조건에 관한 전반적인 명세를 기술한 메타데이터의 집합

관계형 데이터베이스의 한계를 극복하기 위한 데이터 저장소의 새로운 형태

NoSQL의 특징

1. 스키마가 없기에 확장성이 용이

2. 데이터 간의 관계를 정의하지 않음

3. RDBMS에 비해 대용량의 데이터를 저장가능

4. 분산형 구조
여러 곳의 서버에 데이터를 분산 저장해 특정 서버에 장애가 발생했을 때도 데이터 유실 혹은 서비스 중지가 발생하지 않음

5. 고정되지 않은 테이블 스키마를 갖음
RDBMS와 달리 테이블의 스키마가 유동적
데이터를 저장하는 칼럼이 각기 다른 이름과 다른 데이터 타입을 갖는 것이 허용

3. JSON과 파이어 베이스

NoSQL의 종류

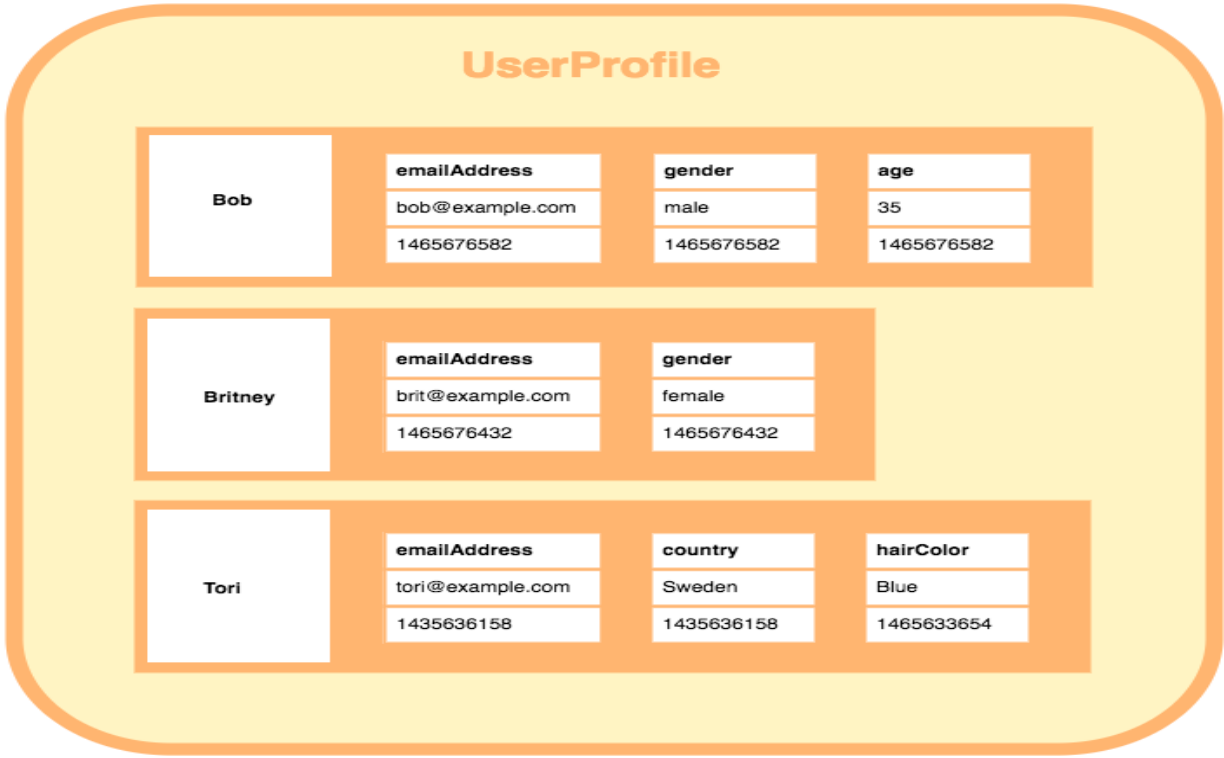
Key-Value Database

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

기본적인 패턴으로 KEY-VALUE 하나의 묶음(Unique)으로 저장되는 구조로 단순한 구조이기에 속도가 빠르며 분산 저장 시 용이

Ex) Redis, Oracle NoSQL Database, VoldeMorte

Wide-Column Database

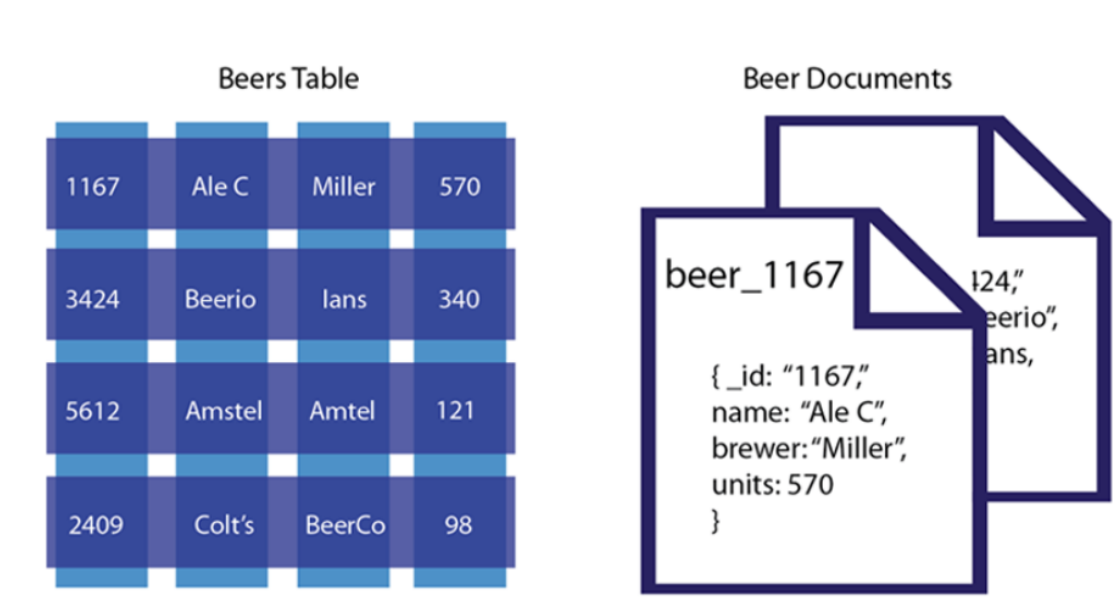


행마다 키와 해당 값을 저장할 때마다 각각 다른 값의 다른 수의 스키마를 가질 수 있음
사용자의 이름(key)에 해당하는 값에 스키마들이 각각 다름을 볼 수 있음
이 데이터베이스는 대량의 데이터의 압축, 분산처리, 집계 쿼리 (SUM, COUNT, AVG 등)및 쿼리 동작 속도 그리고 확장성이 뛰어난 것이 특징
EX) Hbase, GoogleBigTable, Vertica

3. JSON과 파이어 베이스

NoSQL의 종류

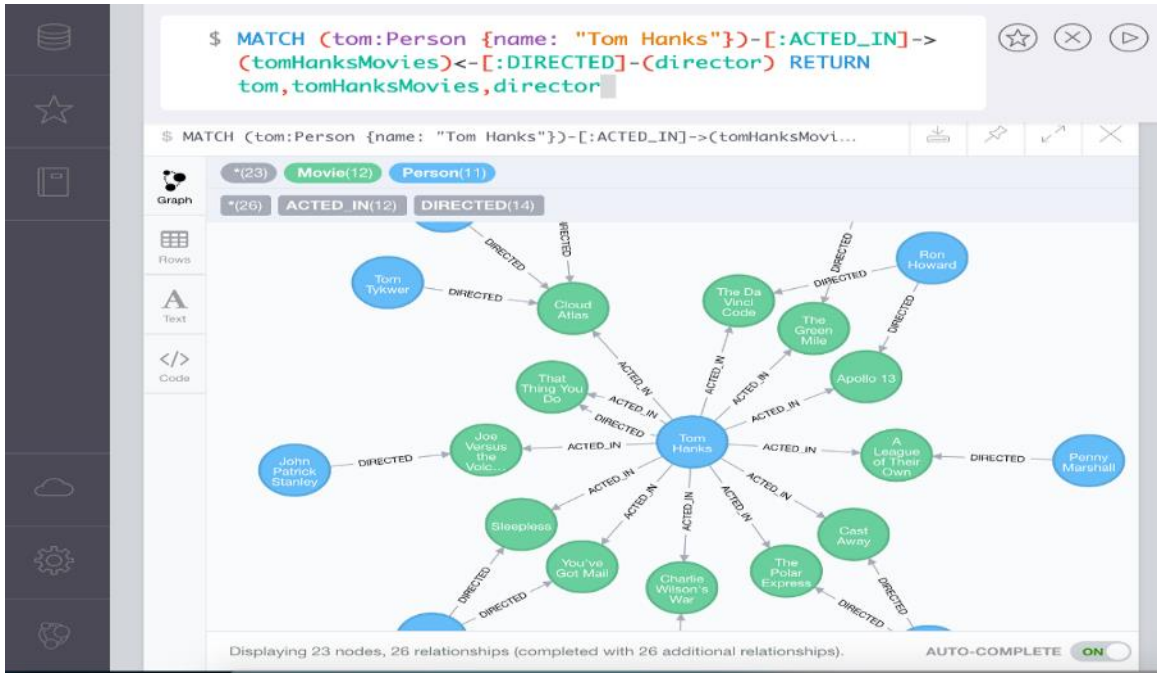
Document Database



테이블의 스키마가 유동적, 즉 레코드마다 각각 다른 스키마를 가질 수 있음
보통 XML, JSON과 같은 DOCUMENT를 이용해 레코드를 저장
트리형 구조로 레코드를 저장하거나 검색하는 데 효과적

Ex) MongoDB, CouchDB, Azure Cosmos DB

Graph Database



데이터를 노드로(그림에서 파란, 녹색 원) 표현하며 노드 사이의 관계를 엣지(그림에서 화살표)로 표현
일반적으로 RDBMS 보다 성능이 좋고 유연하며 유지보수에 용이한 것이 특징.
Social networks, Network diagrams 등에 사용가능

Ex) Neo4j, BlazeGraph, OrientDB

3. JSON과 파이어 베이스

NoSQL인 JSON과 파이어 베이스

NoSQL장점

•RDBMS에 비해 저렴한 비용으로 분산처리와 병렬 처리 가능

•비정형 데이터 구조 설계로 설계 비용 감소

•Big Data 처리에 효과적

•가변적인 구조로 데이터 저장이 가능

NoSQL단점

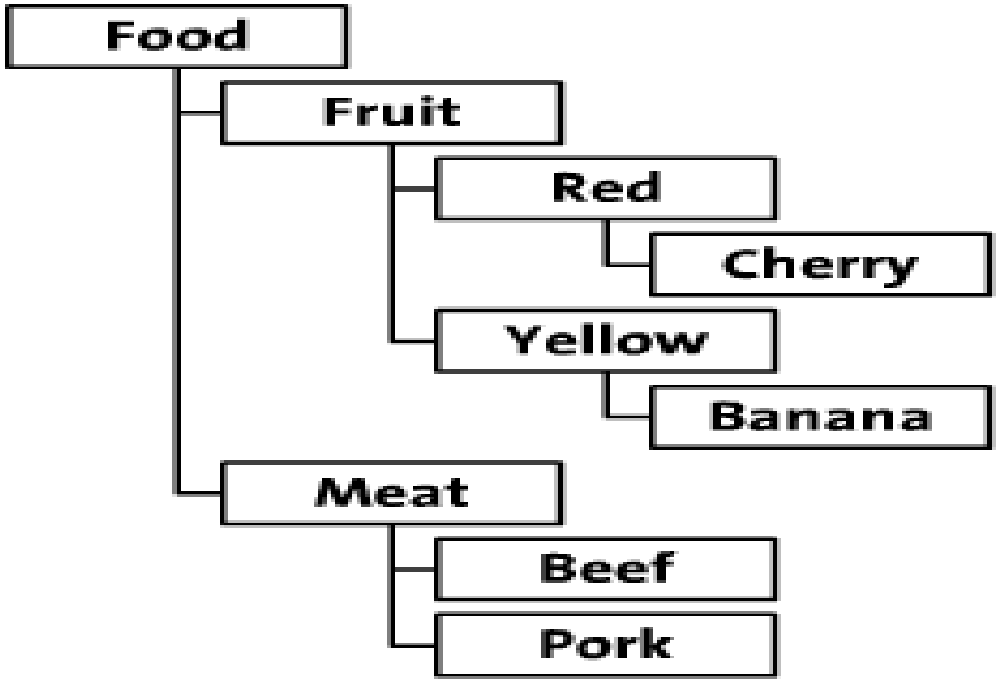
•데이터 업데이트 중 장애가 발생하면 데이터 손실 발생 가능

•많은 인덱스를 사용하려면 충분한 메모리가 필요. 인덱스 구조가 메모리에 저장

•데이터 일관성이 항상 보장되지 않음

JSON과 파이어 베이스

파이어 베이스는 구글에서 제공하는 클라우드 서비스
SQL을 사용하지 않고 JSON방식의 문서를 사용하여 저장
Key-Value를 사용하여 데이터를 주고 받도록 제공
트리 구조로 데이터를 저장



3. JSON과 파이어 베이스

NoSQL인 JSON과 파이어 베이스

게시판에 대한 데이터

게시번호	작성자	제목	본문	작성일	수정일
1	master	[공지]첫 게시글	안녕하세요	20170419000000	20170419000000
2	user1	반갑습니다 ~	제곧내	20170419000000	20170419000000
3	user2	뭐하는 곳일까요?	쓸때없는 글?	20170419000000	20170419000000

이렇게 문서형으로 되어 있기에 JOIN이 불가
JSON은 스키마가 없어 관계를 정의를 해줄 수 없음

게시물 1번에 댓글을 표현하면
이러한 형태

```
{
    "1" :{
        "author" : "master",
        "title"  : "[공지]첫 게시글",
        "body"   : "안녕하세요",
        "createTime" : 20170419000000,
        "editTime" : 20170419000000,
        "reply" : {
            "1" : {
                "author" : "user3",
                "text" : "안녕하세요 영자님",
                "time" : 2017042140000
            }
        }
    }
}
```

JSON 형태

```
{
    "Board" : {
        "1" :{
            "author" : "master",
            "title"  : "[공지]첫 게시글",
            "body"   : "안녕하세요",
            "createTime" : 20170419000000,
            "editTime" : 20170419000000
        },
        "2" : {
            "author" : "user1",
            "title"  : "[반갑습니다 ~",
            "body"   : "제곧내",
            "createTime" : 20170419000000,
            "editTime" : 20170419000000
        },
        "3" : {
            "author" : "user2",
            "title"  : "뭐하는 곳일까요?",
            "body"   : "쓸때없는 글?",
            "createTime" : 20170419000000,
            "editTime" : 20170419000000
        }
    }
}
```

3. JSON과 파이어 베이스

JSON개요

JSON이란?

JSON은 JavaScript Object Notation의 약자
경량(Lightweight)의 DATA 교환 형식
사람이 읽을 수 있는 텍스트 기반의 데이터 교환 표준

JSON은 XML의 대안으로서 좀 더 쉽게 데이터를 교환하고 저장하기 위하여 고안되었고 텍스트 기반이므로 어떠한 프로그래밍 언어에서도 JSON 데이터를 읽고 사용가능

사람과 기계 모두 이해하기 쉬우며 용량이 작아서, 최근에는 JSON이 XML을 대체해서 데이터 전송 등에 많이 사용

OPEN API 대부분 JSON을 활용하여 데이터를 주고받음

3. JSON과 파이어 베이스

JSON의 특징

JSON은 자바스크립트를 확장하여 만들어짐

JSON은 자바스크립트 객체 표기법을 따름

JSON은 사람과 기계가 모두 읽기 편하도록 고안

JSON은 프로그래밍 언어와 운영체제에 독립적

3. JSON과 파이어 베이스

JSON의 구조

JSON의 구조

1. JSON 데이터는 이름과 값의 쌍으로 구성
2. JSON 데이터는 쉼표(,)로 나열
3. 객체(object)는 중괄호({})로 둘러쌓아 표현
4. 배열(array)은 대괄호([])로 둘러쌓아 표현

문법

```
"데이터이름": 값
```

JSON 객체

```
{
    "name": "식빵",
    "family": "웰시코기",
    "age": 1,
    "weight": 2.14
}
```

JSON 배열

```
"dog": [
    {"name": "식빵", "family": "웰시코기", "age": 1, "weight": 2.14},
    {"name": "콩콩", "family": "포메라니안", "age": 3, "weight": 2.5},
    {"name": "젤리", "family": "푸들", "age": 7, "weight": 3.1}
]
```

3. JSON과 파이어 베이스

JSON의 타입

JSON의 타입

1. 숫자(number)
2. 문자열(string)
3. 불리언(boolean)
4. 객체(object)
5. 배열(array)
6. null

규칙

- 데이터이름은 문자열로 반드시 큰따옴표("")사용
- 배열은 JSON에서 제공하는 기본 타입을 모두 저장가능
- JSON에서 null이란 값을 가지고 있지 않다는 의미를 가지는 하나의 데이터 값

- 객체와 배열의 차이
- 배열과 객체는 여러 데이터를 묶어 놓은 집합이라는 점에서 서로 비슷한 타입
- 하지만 객체는 프로퍼티의 집합이며, 배열은 데이터 값의 집합

숫자

```
{
    "age": 1
}
```

문자열과 불린언

```
{
    "name": "식빵",
    "lunch": true
}
```

객체

```
{
    "dog": {
        "name": "식빵",
        "family": "웰시코기",
        "age": 1,
        "weight": 2.14,
        "owner": {
            "ownerName": "홍길동",
            "phone": "01012345678"
        }
    }
}
```

배열

```
{
    "dog": [
        "웰시코기",
        "포메라니안",
        "푸들",
        {
            "ownerName": "홍길동",
            "phone": "01012345678"
        }
    ]
}
```

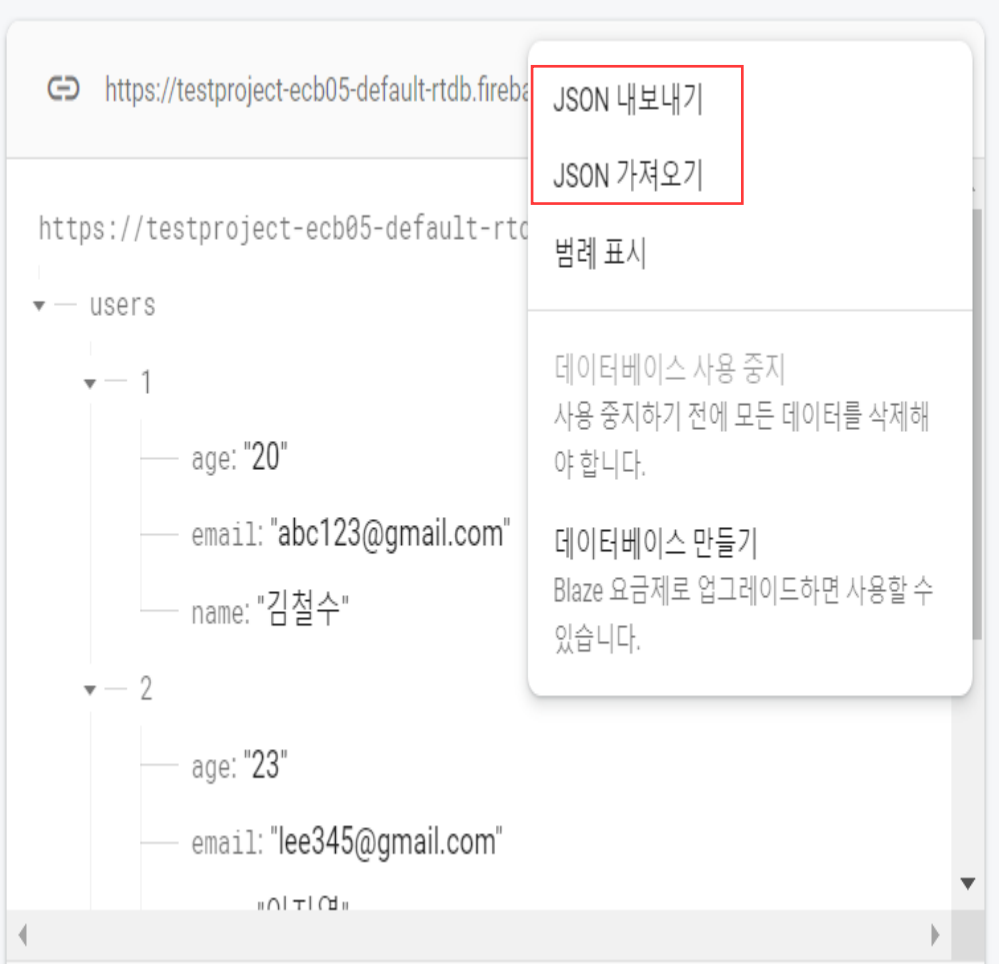
null

```
{
    "id": 1,
    "name": null
}
```

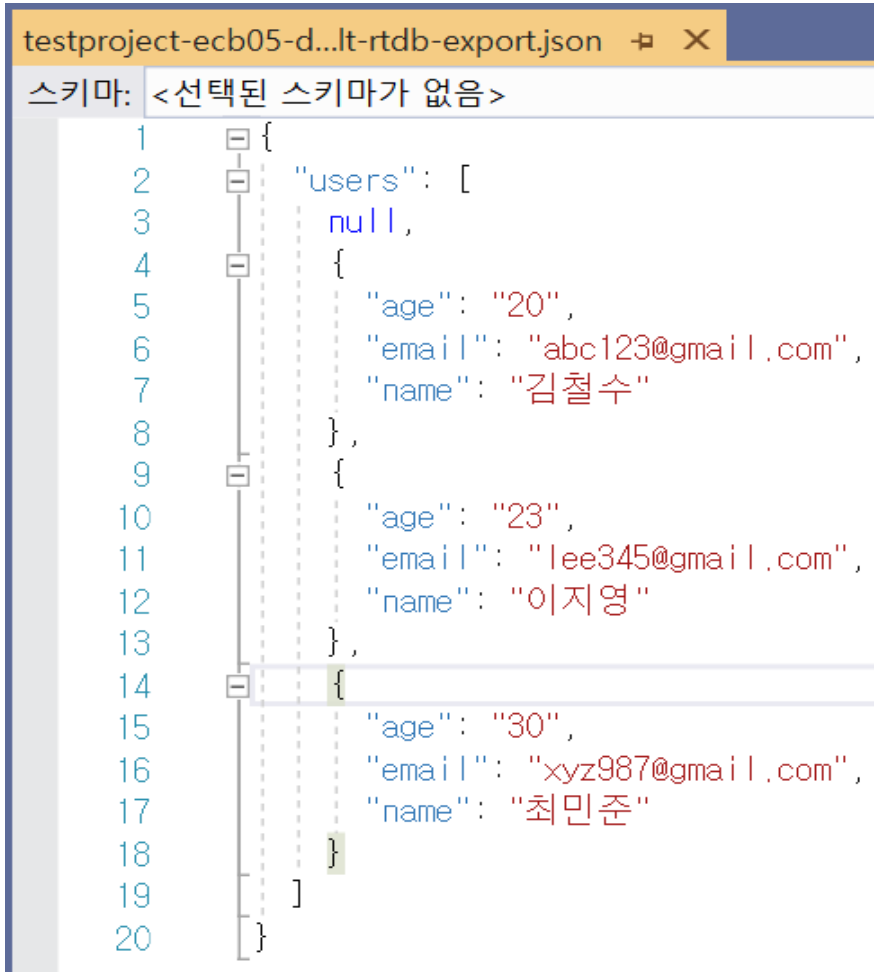
3. JSON과 파이어 베이스

파이어 베이스에서 JSON 관리

JSON파일을 관리가 간편



JSON형태로 데이터 가져오기



3. JSON과 파이어 베이스

파이어 베이스에서 JSON 관리

엑셀이나 메모장으로 JSON파일 작성

test.json

스키마: <선택된 스키마가 없음>

```json
{
  "Board": {
    "1": {
      "author": "master",
      "title": "[공지]첫 게시글",
      "body": "안녕하세요",
      "createTime": 20220527000000,
      "editTime": 20220527000000
    },
    "2": {
      "author": "user1",
      "title": "반가워요!",
      "body": "제곧내",
      "createTime": 20220527000000,
      "editTime": 20220527000000
    },
    "3": {
      "author": "user2",
      "title": "뭐하는 곳일까요?",
      "body": "아무말 대잔치",
      "createTime": 20220527000000,
      "editTime": 20220527000000
    }
  }
}
```

JSON 데이터 가져오기
기존 데이터는 삭제

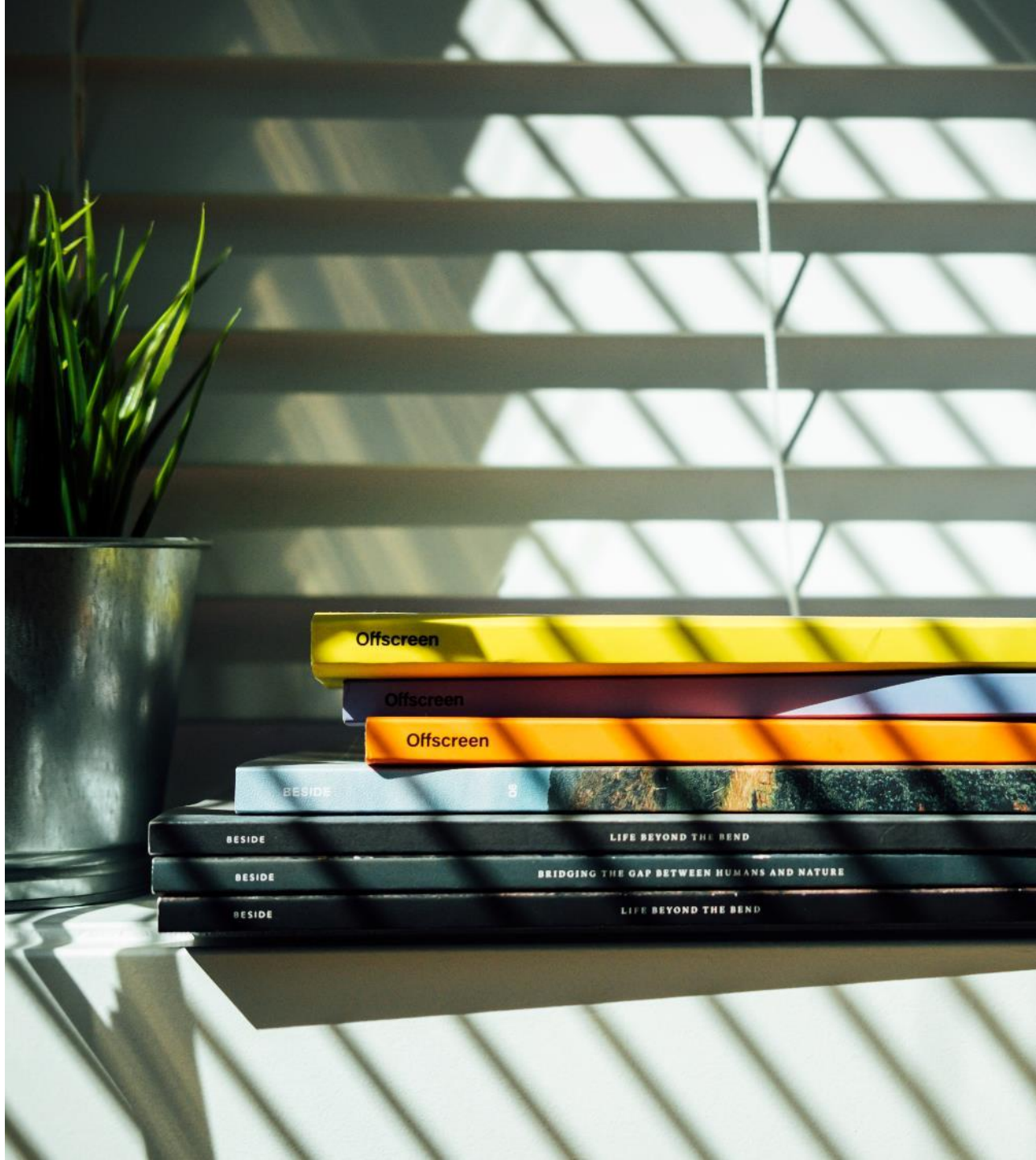
https://testproject-ecb05-default-rtdb.firebaseio.com

https://testproject-ecb05-default-rtdb.firebaseio.cor

- Board
  - 1
    - author: "master"
    - body: "안녕하세요"
    - createTime: 20220527000000
    - editTime: 20220527000000
    - title: "[공지]첫 게시글"
  - 2
    - author: "user1"
    - body: "제곧내"
    - createTime: 20220527000000
    - editTime: 20220527000000
    - title: "반가워요!"

Part 4,

소프트웨어 공학



4. 소프트웨어 공학

소프트웨어 개요

소프트웨어란?

1. 응용 프로그램과 데이터처럼, 컴퓨터의 하드웨어 상에서 구동되거나 처리되는 무형물을 지칭하는 말

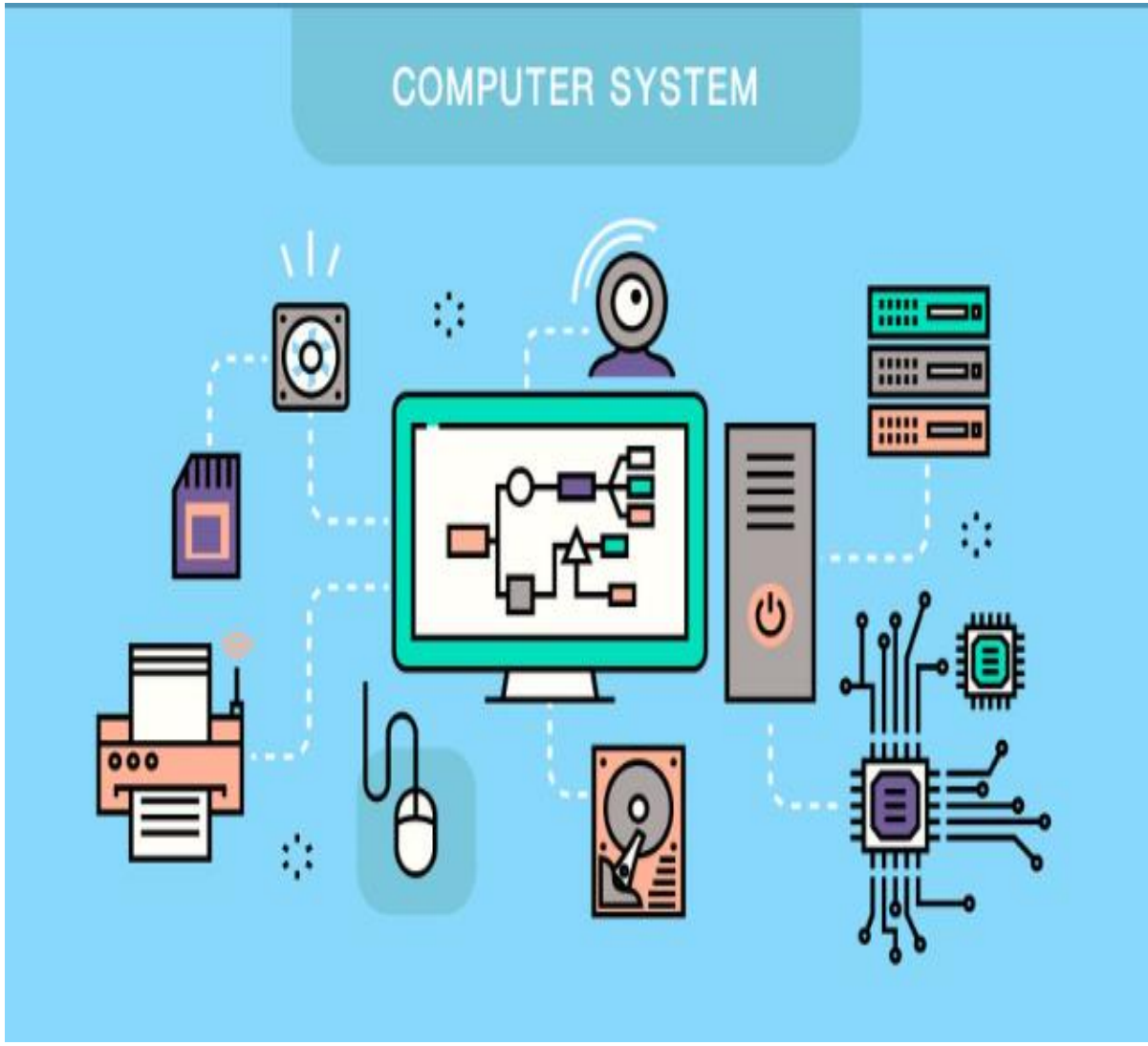
2. 소프트웨어는 컴퓨터의 시스템을 구성하는 주요 요소 중 하나(컴퓨터 = 하드웨어 + 소프트웨어 + 펌웨어)

펌웨어란? 전자기기 등의 기본적인 제어 및 구동을 맡는 소프트웨어

3. 원시코드, 모든 산출물(자료구조, DB구조, 테스트 결과 등), 각 단계마다 생성 되는 문서, 사용자 매뉴얼 등이 존재

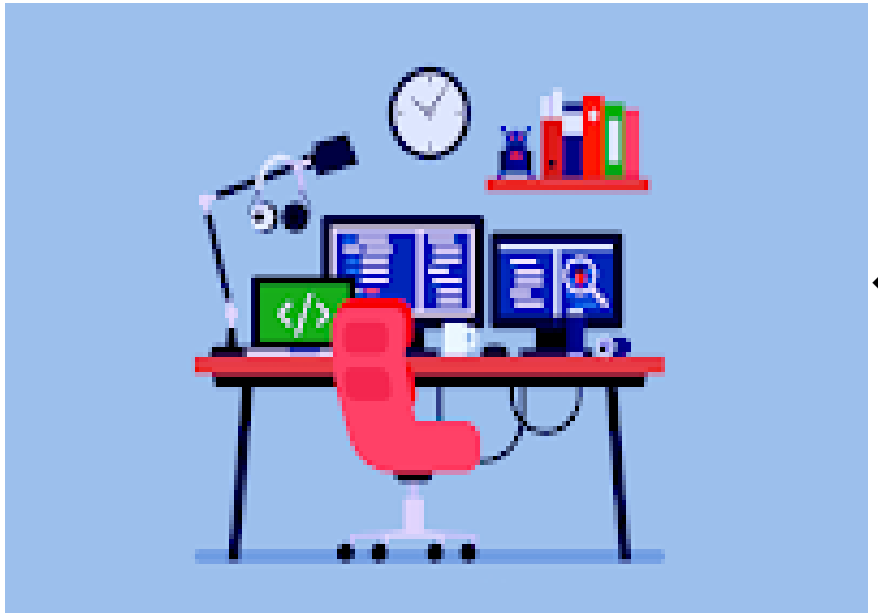
4. 소프트웨어는 프로그램 뿐만 아니라 그 이상의 것도 포함하는 매우 포괄적인 개념(소프트웨어 >> 프로그램)

하드웨어는 신체, 소프트웨어는 뇌(정신)



4. 소프트웨어 공학

소프트웨어 특징



소프트웨어 특징

❖ 제조가 아닌 개발이 필요

◆ 제조 : 정해진 틀에 맞춰 일정하게 생산하는 것(공장)
인력이 많이 필요하고 능력별 결과물 차이가 근소

◆ 개발 : 새로운 것을 창조하는 것
개인 능력별 결과물 차이가 매우 큼

❖ 소모가 아닌 품질 저하

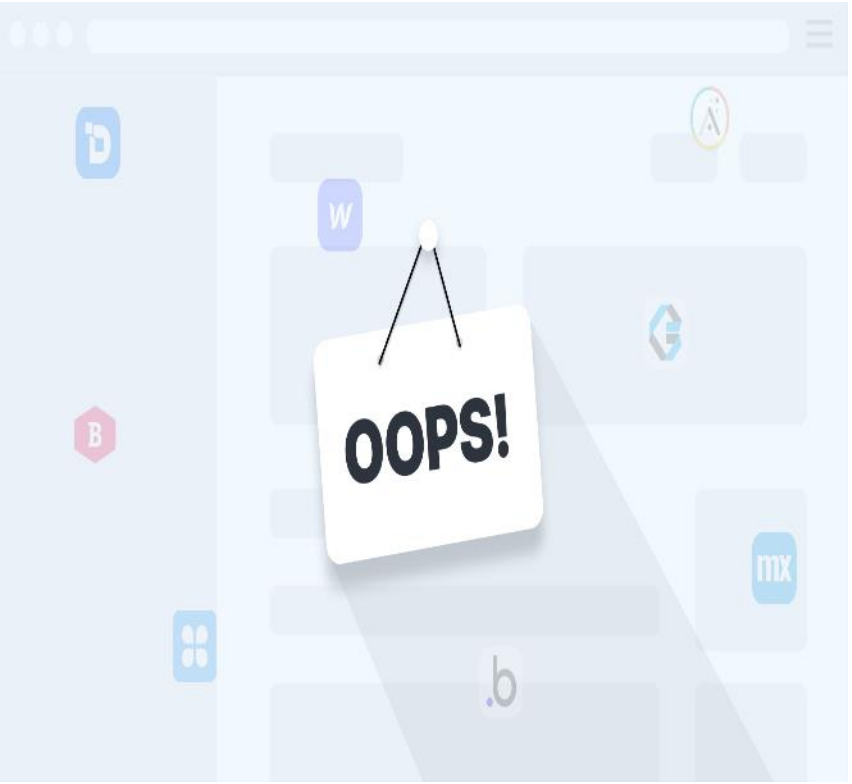
◆ H/W : 오래 사용하면 부품이 닳고, 고장 발생 빈도 증가, 기능도 떨어짐

◆ S/W : 오래 사용해도 닳지 않고, 고장 발생 빈도 낮고, 기능도 동일
처음부터 만들어가는 것이기에 근본적인 차이로 발전 속도 느림

※ 해결방안으로 CBD개발 방법론이 존재
컴포넌트를 조합해 재사용함으로써 개발 생산성과 품질을 높이고 시스템 유지보수 비용을 최소화할 수 있는 개발방법론

4. 소프트웨어 공학

소프트웨어 개발의 어려움



소프트웨어 개발의 어려움

- 명세화의 어려움 : 보고 사용해보기 전까지는 필요한 것을 정의하기 어려움
- 재사용의 어려움 : 전에 만들어 놓은 컴포넌트를 다른 소프트웨어에 적용할 수 없음
- 예측의 어려움 : 외부 요소(업무 절차, 법규, 하드웨어, 데이터 형식 등)의 영향으로 예측이 어려움
- 유지보수의 어려움 : 다른 사람이 개발한 소프트웨어를 쉽게 이해하기 어려움
- 고품질의 어려움 : 한 줄의 코드를 수정하더라도 테스트할 내용이 많아짐

4. 소프트웨어 공학

소프트웨어 공학이란?

정의

- 공학을 소프트웨어에 적용하는 것
- 소프트웨어의 개발, 운용, 유지보수 등의 생명 주기 전반을 체계적이고 서술적이며 정량적으로 다루는 학문

목적

1. 복잡도 저하 : 큰 문제를 작은 문제로 나누는 방법 제공
2. 비용 최소화
3. 개발 기간 단축
4. 대규모 프로젝트 관리
5. 고품질 소프트웨어 생산
6. 효율적 개발을 통한 생산성을 향상

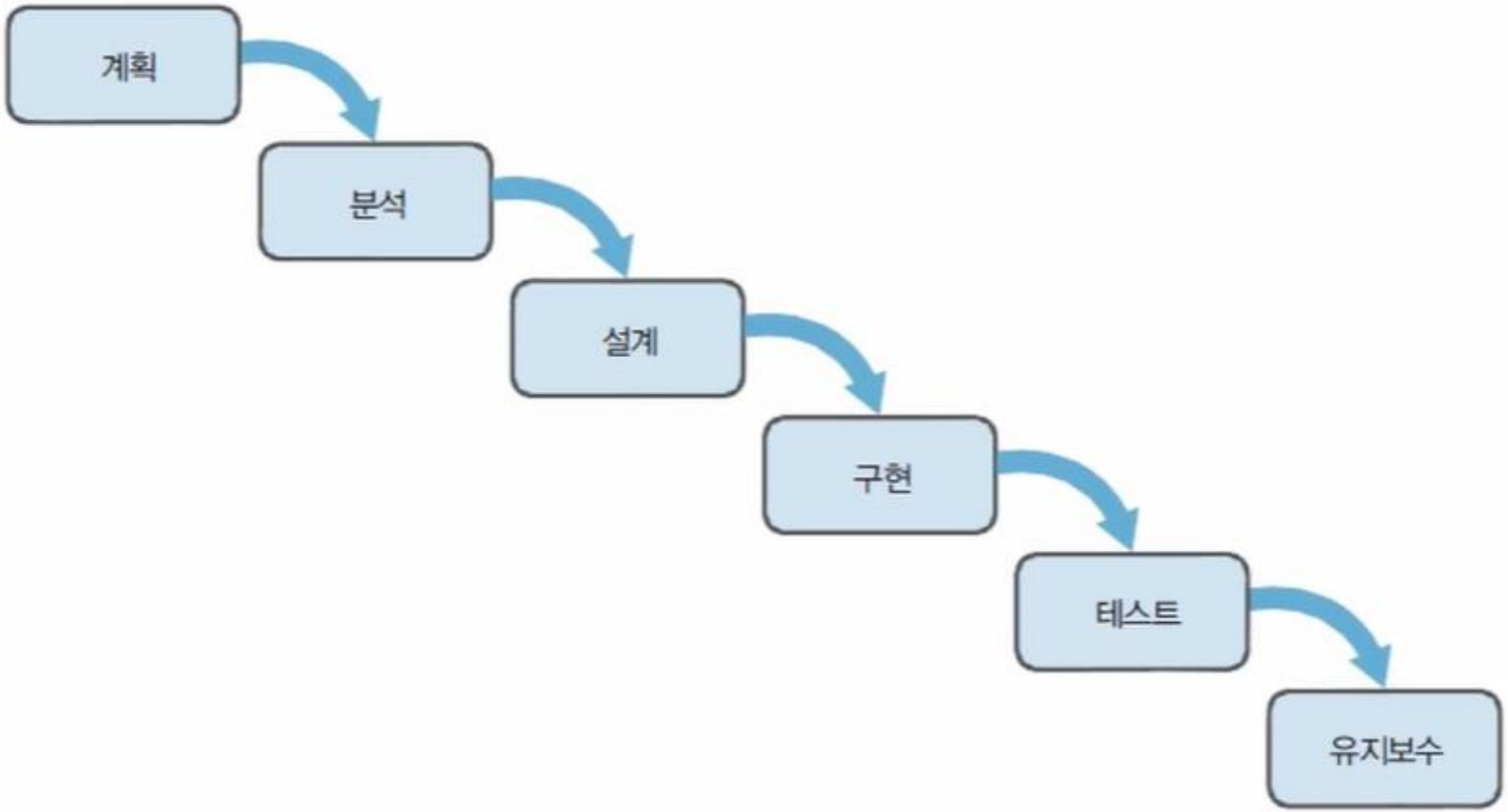
소프트웨어 공학은 여러 가지 원리와 방법을 **적용하여 품질 좋은 소프트웨어**를 **최소의 비용**으로 **계획된 일정에 맞추어** 개발하는 것

4. 소프트웨어 공학

소프트웨어 개발 생명주기

소프트웨어 개발 생명주기(SDLC Software Development Life Cycle)

계획 단계에서 유지보수 단계에 이르기까지 일어나는 일련의 과정



4. 소프트웨어 공학

소프트웨어 개발 생명주기

1단계 : 계획

인건비 및 재료비 계산을 비롯해 프로젝트 조건을 평가하여 목표가 설정된 일정표를 구성

✓ 개발 비용 산정

- COCOMO모델
  - 비용 추정 모델로서 소프트웨어 개발비, 유지보수 비용의 견적, 개발 단계 및 업무 활동과 산출물을 정의하여 포함한 비용 산정
  - Basic COCOMO, Intermediate COCOMO, Detailed COCOMO로 분류
- 기능점수(FP)모델
  - 소프트웨어의 양과 질을 동시에 고려한 소프트웨어 규모산정 방식

✓ 일정 계획

- 작업분할구조도 WBS(Work Breakdown Structure)
  - 프로젝트 목표 달성을 위하여 필요한 업무들과 액티비티들을 세분화 하는 작업
- 간트 차트(Gantt chart)
  - 프로젝트 일정 관리를 위한 바(bar) 형태의 도구

4. 소프트웨어 공학

소프트웨어 개발 생명주기

2단계 : 분석

- 개발할 소프트웨어의 기능과 제약조건, 목표 등을 실제 사용자와 함께 명확히 정의하는 단계
- 개발 방법과 필요한 자원 및 예산 예측 후 요구명세를 작성
- 기능 / 비기능 요구사항으로 구분

기능 요구사항

개념 - 시스템이 제공하는 기능, 서비스에 대한 요구사항
특징 - 기능성, 완전성, 일관성
Ex) 쇼핑몰 장바구니 기능, 무통장 입금 등

비기능 요구사항

개념 - 시스템이 수행하는 기능 이외의 사항, 시스템 구축에 대한 제약사항에 관한 요구사항
특징 – 신뢰성, 사용성, 효율성, 유지보수성, 이식성, 보안성 등
Ex) 특정 함수 호출은 3초 초과 금지 등

4. 소프트웨어 공학

소프트웨어 개발 생명주기

3단계 : 설계

✓ 시스템 요구사항 단계에서 정의한 기능을 실제 수행할 수 있도록 수행 방법을 논리적으로 결정하는 단계
✓ 시스템 구조 설계/ 프로그램 설계 / 사용자 인터페이스 설계로 구성

- 시스템 구조 설계
  - 시스템을 구성하는 내부 프로그램이나 모듈 간의 관계와 구조를 설계
- 프로그램 설계
  - 프로그램 내의 각 모듈에서의 처리 절차나 알고리즘을 설계
- 사용자 인터페이스 설계
  - 사용자가 시스템을 사용하기 위해 보이는 부분을 설계

설계 원리 : 분할과 정복, 추상화, 단계적 분해, 모듈화, 정보은닉

- 분할과 정복
  - 어려운 큰 문제를 작고 쉬운 문제로 분할하여 정복

- 추상화
  - 고수준 : 인간에 친숙(자판기 사용자)
  - 저수준 : 기계에 가까움(자판기 수리 기사)

4. 소프트웨어 공학

소프트웨어 개발 생명주기

4단계 : 구현

- 설계 단계에서 논리적으로 결정한 문제 해결 방법을 특정 프로그래밍 언어를 사용하여 구현하는 단계프로그래밍 언어 선택, 기법, 스타일, 순서 등을 결정하는 단계
- 인터페이스 개발 / 자료 구조 개발 / 오류 처리로 구성
- 프로그래밍 기법은 구조화 프로그래밍과 모듈러 프로그래밍

•구조화 프로그래밍

- 조건문, 반복문을 사용하여 프로그램을 작성하고, 순차구조, 선택구조, 반복구조의 세 가지 제어구조로 표현

장점

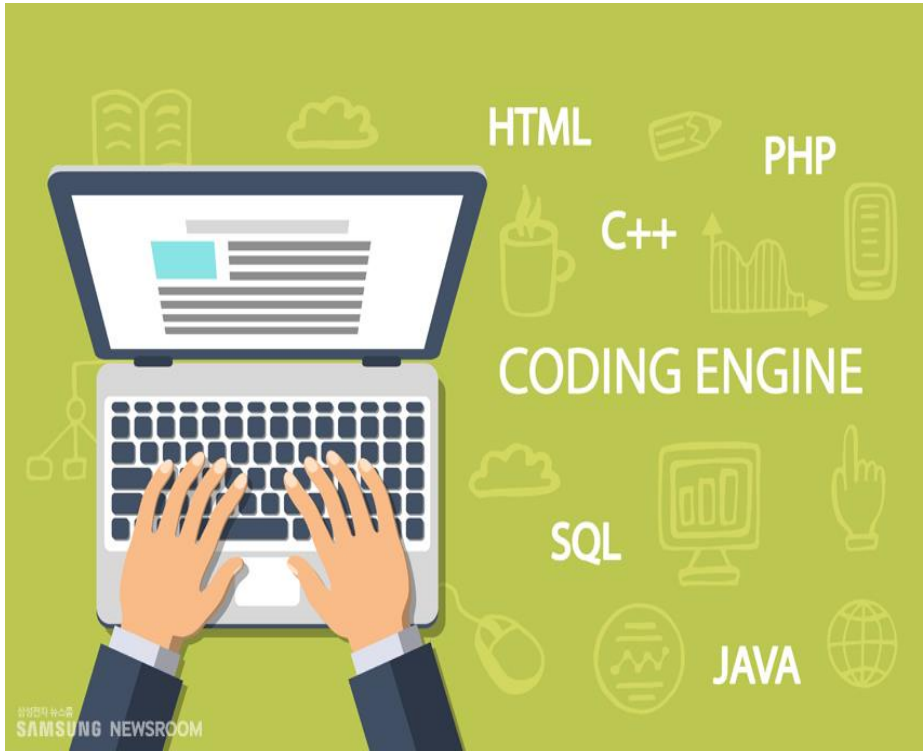
- 구조가 명확하여 정확성 검증과 테스트 및 유지보수가 쉬움

•모듈러 프로그램

- 프로그램을 여러 개의 작은 모듈로 나누어 계층 관계로 구성하는 프로그래밍 기법

장점

- 모듈별로 개발과 테스트 유지 및 유지 보수 가능하며, 모듈의 재사용 가능



4. 소프트웨어 공학

소프트웨어 개발 생명주기

5단계 : 테스트

•시스템이 요구를 만족하는지, 예상과 실제 결과가 어떤 차이를 보이는지 검사하고 평가하는 단계
•단위 테스트 / 통합 테스트 / 시스템 테스트 / 인수 테스트로 구분

- **단위 테스트(Unit Test)**
  - 사용자 요구사항에 대한 단위 모듈, 서브루틴 등을 테스트하는 단계
  - 기법 - 자료 구조 테스트, 실행 경로 테스트, 오류 처리 테스트, 인터페이스 테스트

- **통합 테스트(Integration Test)**
  - 단위 테스트를 통과한 모듈 사이의 인터페이스로써 통합된 컴포넌트 간의 상호작용을 검증하는 단계
  - 기법 - 빅뱅 테스트, 샌드위치 테스트, 상향식 테스트, 하향식 테스트

- **시스템 테스트(System Test)**
  - 통합된 단위 시스템의 기능이 시스템에서 정상적으로 수행되는지를 검증하는 단계
  - 기법 – 기능, 비기능 요구사항 테스트

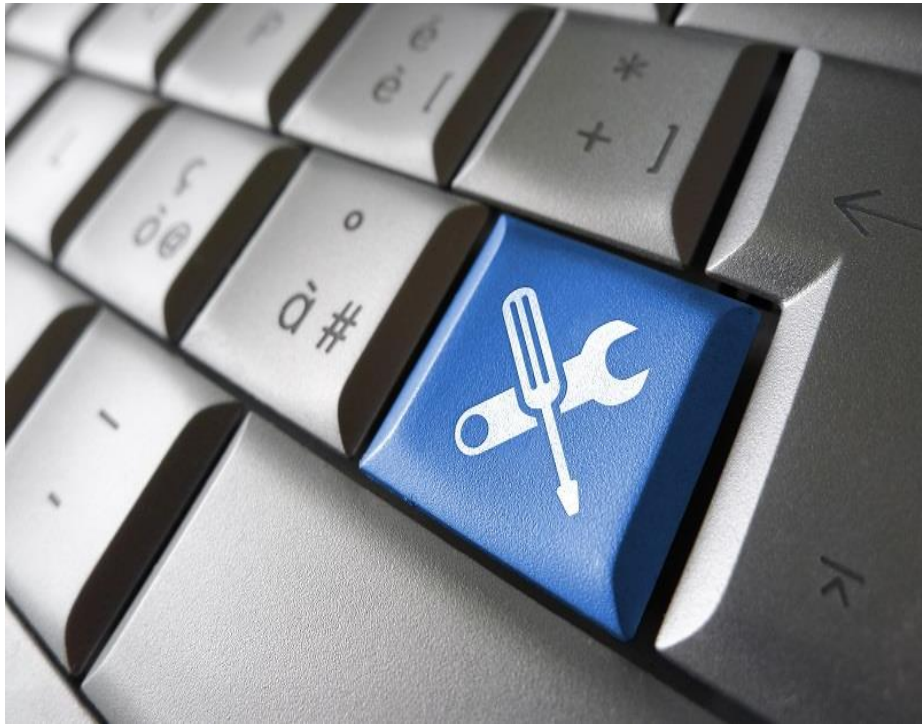
- **인수 테스트(Acceptance Test)**
  - 계약상의 요구사항이 만족되었는지 확인하기 위한 테스트 단계
  - 기법 - 계약 인수, 규정 인수, 사용자 인수, 운영상의 인수, 알파, 베타 테스트

4. 소프트웨어 공학

소프트웨어 개발 생명주기

6단계 : 유지 보수

✓ 시스템이 인수되고 설치된 후 일어나는 모든 활동
✓ 이후 일어나는 커스터마이징, 구현, 테스트 등 모두 이 단계 에 포함되므로 가장 긴 기간을 차지
✓ 유지보수의 유형에는 수정형, 적응형, 완전형, 예방형



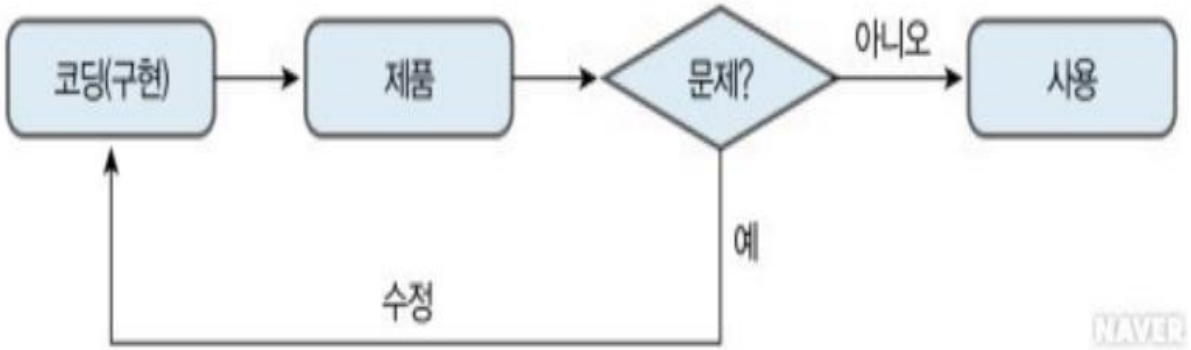
- 수정형 유지보수
  - 사용 중에 발견한 프로그램의 오류 수정 작업을 진행
- 적응형 유지보수
  - 시스템과 관련한 환경적 변화에 적응하기 위한 재조정 작업
- 완전형 유지보수
  - 시스템의 성능을 향상하기 위한 개선 작업
- 예방 유지보수
  - 앞으로 발생할지 모를 변경 사항을 수용하기 위한 대비 작업을 수행

4. 소프트웨어 공학

소프트웨어 개발 프로세스 모델

1. 주먹구구식 모델

요구 분석 명세서나 설계 단계 없이 개발
코드를 작성해 제품을 만들고나서 설계,유지보수 진행



단점
✓ 산출물이 없어 관리 및 유지보수 어려움
✓ 범위를 알 수 없고 좋은 구조를 만들 수 없음
✓ 나누어 개발 할 수가 없음
✓ 진척 상황 파악 어려움
✓ 계속 수정을 하면 구조가 나빠져 수정이 어려워 짐

4. 소프트웨어 공학

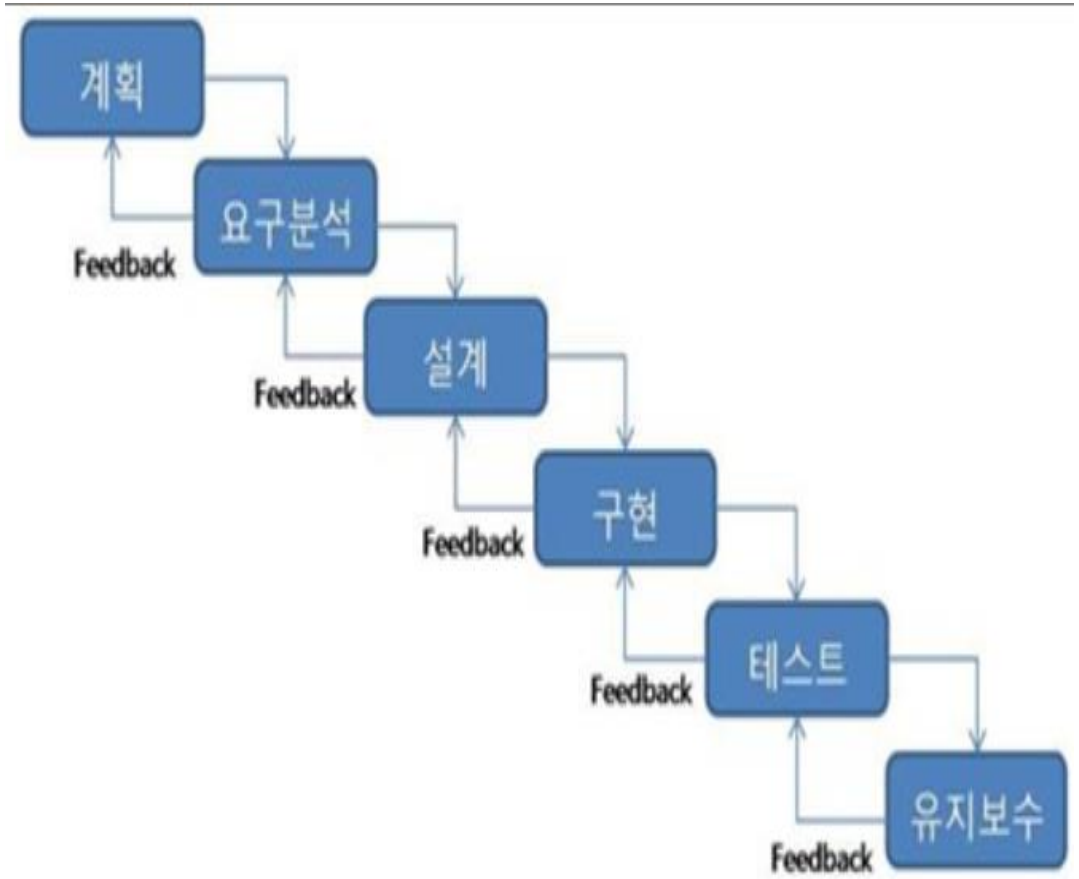
소프트웨어 개발 프로세스

소프트웨어 개발 프로세스 모델

2. 선형 순차적 모델(Waterfall Model 폭포수 모델)

한번 떨어진 물은 다시 올라가는 일이 없듯이 각 단계를 완벽하게 마무리 한 후에 다음 단계로 넘어가는 모델

- 각 단계가 마무리 된 후, 사용자에게 확인을 받음
- 제품의 일부가 될 매뉴얼을 작성해야 함
- 2개 이상의 과정이 병행하며 수행 되지 않음
- 각 단계가 끝난 후에는 결과물이 정확하게 산출되어야 함
- 요구사항 변경이 어려움



장점

- 단계별 산출물이 정확하여 개발 공정의 기준점을 제시
- 모형의 적용 경험과 성공 사례가 많음
- 단계별 정의가 분명하고, 전체 공조의 이해가 용이

단점

- 앞단계가 완료되어야 넘어갈 수 있음
- 결과물이 완벽해야만 다음단계에서 오류를 넘기지 않음
- 중간에 가시적인 결과를 볼 수 없음

4. 소프트웨어 공학

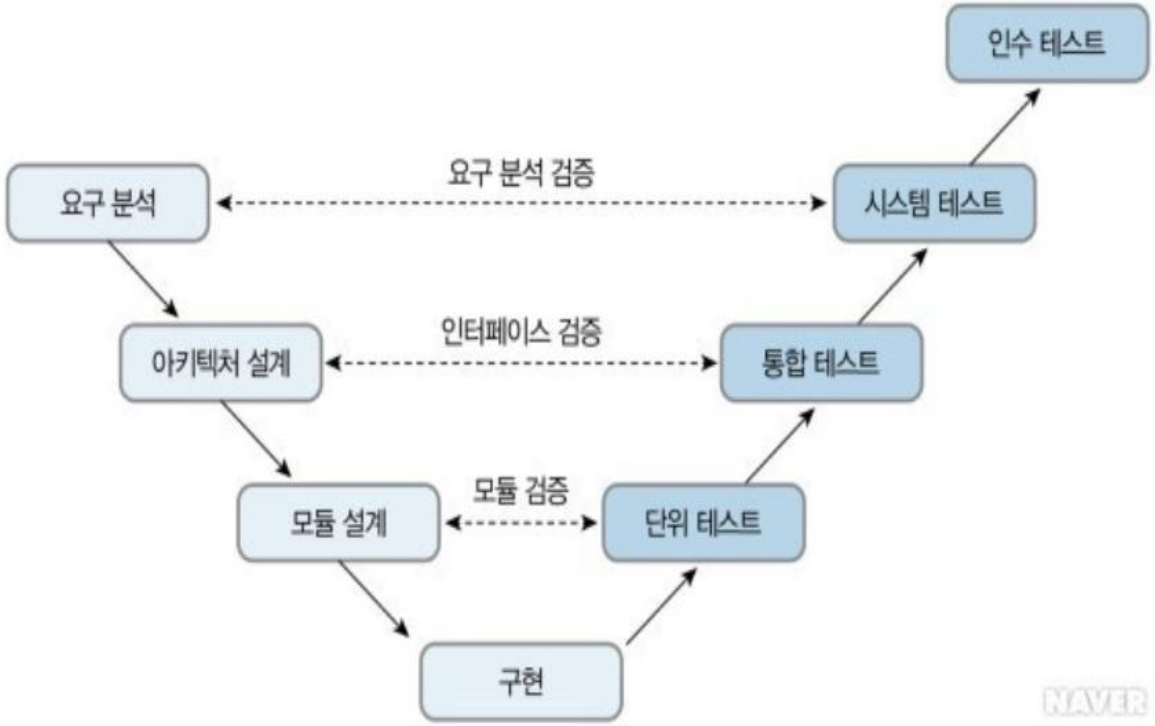
소프트웨어 개발 프로세스

소프트웨어 개발 프로세스 모델

3. V모델

폭포수모델 + 테스트 단계 추가 확장

각 단계마다 테스트를 추가하여 추후 프로그램 사용 시에 발생할 수 있는 오류를 제작 과정에서 찾아 없애자는 목표를 가짐



- 상세한 문서화를 통해 작업을 진행하는 방법을 사용
- 테스트 활동을 코딩 이후가 아닌 프로젝트 시작 시에 시작
- 많은 양의 프로젝트 비용과 시간이 감소

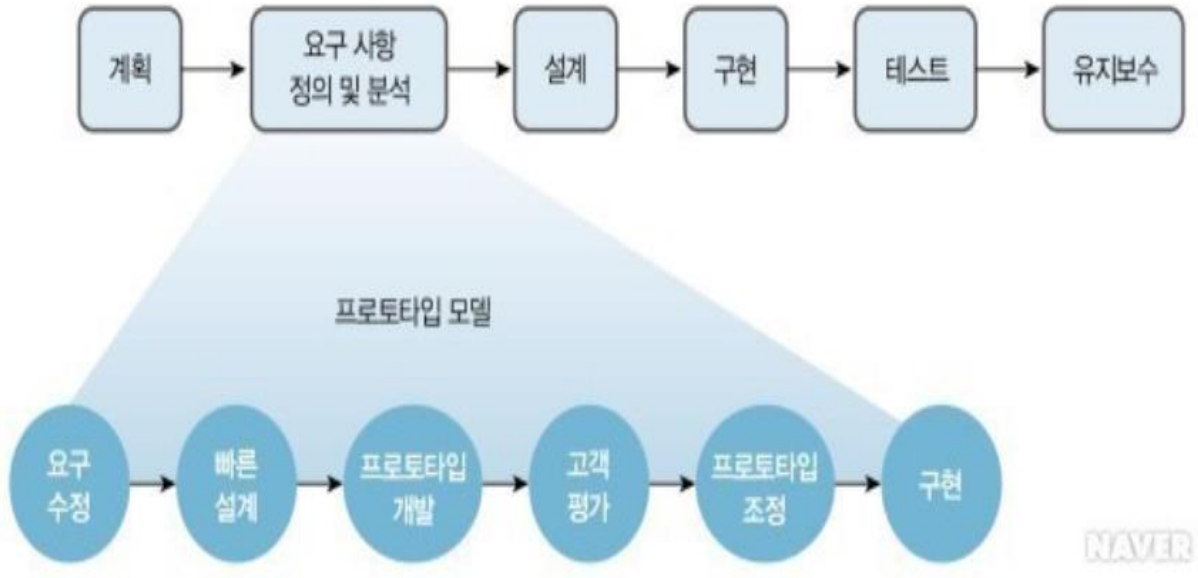
4. 소프트웨어 공학

소프트웨어 개발 프로세스

4. 진화적 프로세스 모델(프로토타입 모델)

사용자의 요구사항을 정확하게 파악하기 위해 실제 개발될 소프트웨어에 대한 견본품을 만들어 최종 결과물을 예측하는 모형

프로토타입 : 미리 제작 해보는 원형, 시제품, 제작물의 모형(모델하우스)



- 완제품을 개발하는 것이 아님
- 입출력화면을 통한 사용자 인터페이스 중심 설계
- 입력 화면을 통한 사용자의 요구 항목 확인
- 출력 결과를 통해 사용자가 원하는 것인지 확인

장점

- 프로토타입이 의사소통 도구로 활용
- 프로토타입 사용을 통한 완성품 예측 가능

단점

- 반복적 개발을 통한 투입 인력 및 비용 산정의 어려움
- 중간 산출물 생성의 어려움

4. 소프트웨어 공학

소프트웨어 개발 프로세스

소프트웨어 개발 프로세스 모델

5. 나선형 모델(Spiral Model)

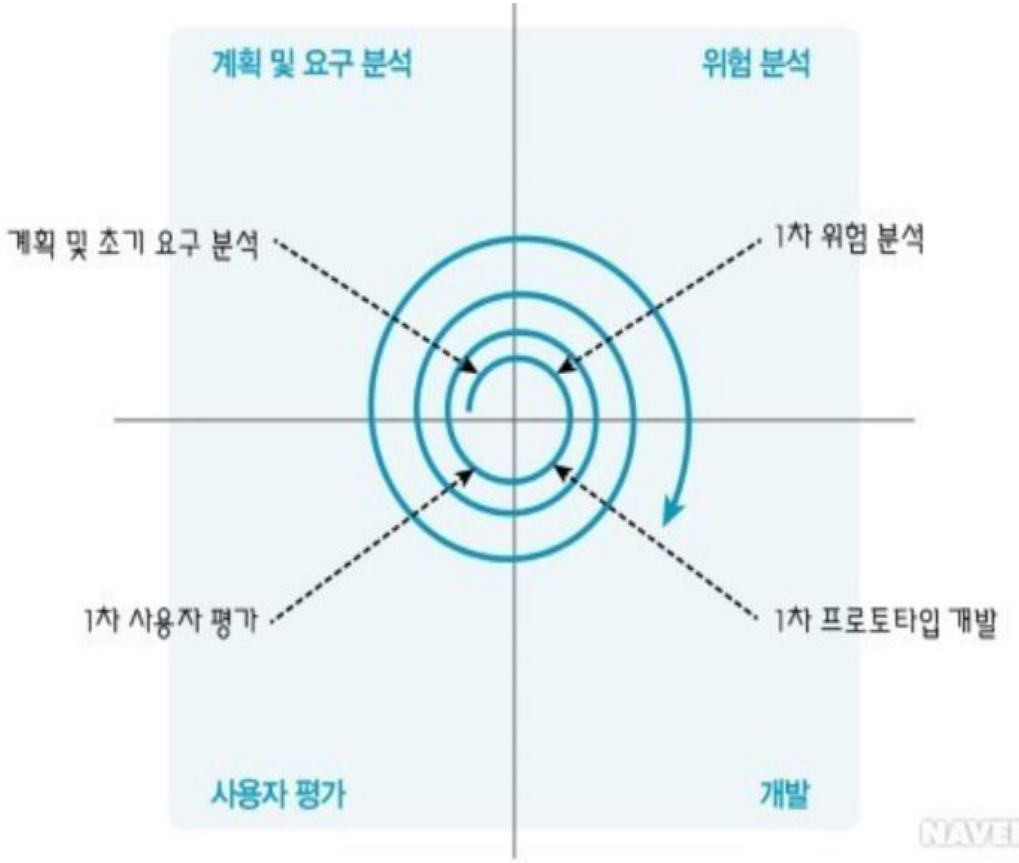
나선형 모델은 폭포수 모델과 진화적 프로세스 모델의 장점이 합쳐진 것

소프트웨어를 개발하면서 발생할 수 있는 위험을 관리하고 최소화하기 위해 점진적으로 완벽한 시스템으로 개발해 나가는 모델

1차 프로토타입 모델은 사용자에게 검증을 받은 후, 추가 요구 사항 및 수정 내용을 참고하여 2차 프로토타입 모델을 완성
다시 사용자에게 검증을 받고 수정하는 과정이 수차례 반복되면서 프로토타입을 벗어나 제품이 완성

장점: 기술과 관리의 위험 요소들을 하나씩 제거해 나감으로써 완성도 높은 소프트웨어를 만들 수 있음

단점 : 위험성 평가에 크게 의존하기 때문에 위험성을 발견하지 못하면 반드시 문제가 발생



Part 5,

깃허브 연동

5. 깃허브 연동

깃과 깃허브

깃(Git)이란?

Git은 형상 관리 도구 중 하나로, 컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템

버전 관리 시스템이란?
문서나 설계도, 소스 코드 등의 변경점을 관리해주는 소프트웨어

장점
소스코드를 따로 주고 받을 필요 없이, git을 사용하면 하나의 프로젝트, 같은 파일을 여러 사람이 동시에 작업하는 병렬 개발이 가능

깃허브(GitHub)란?

깃허브(GitHub)는 분산 버전 관리 툴인 깃(Git)을 기반으로 소스 코드를 호스팅하고, 협업 지원 기능들을 지원하는 마이크로소프트의 웹서비스

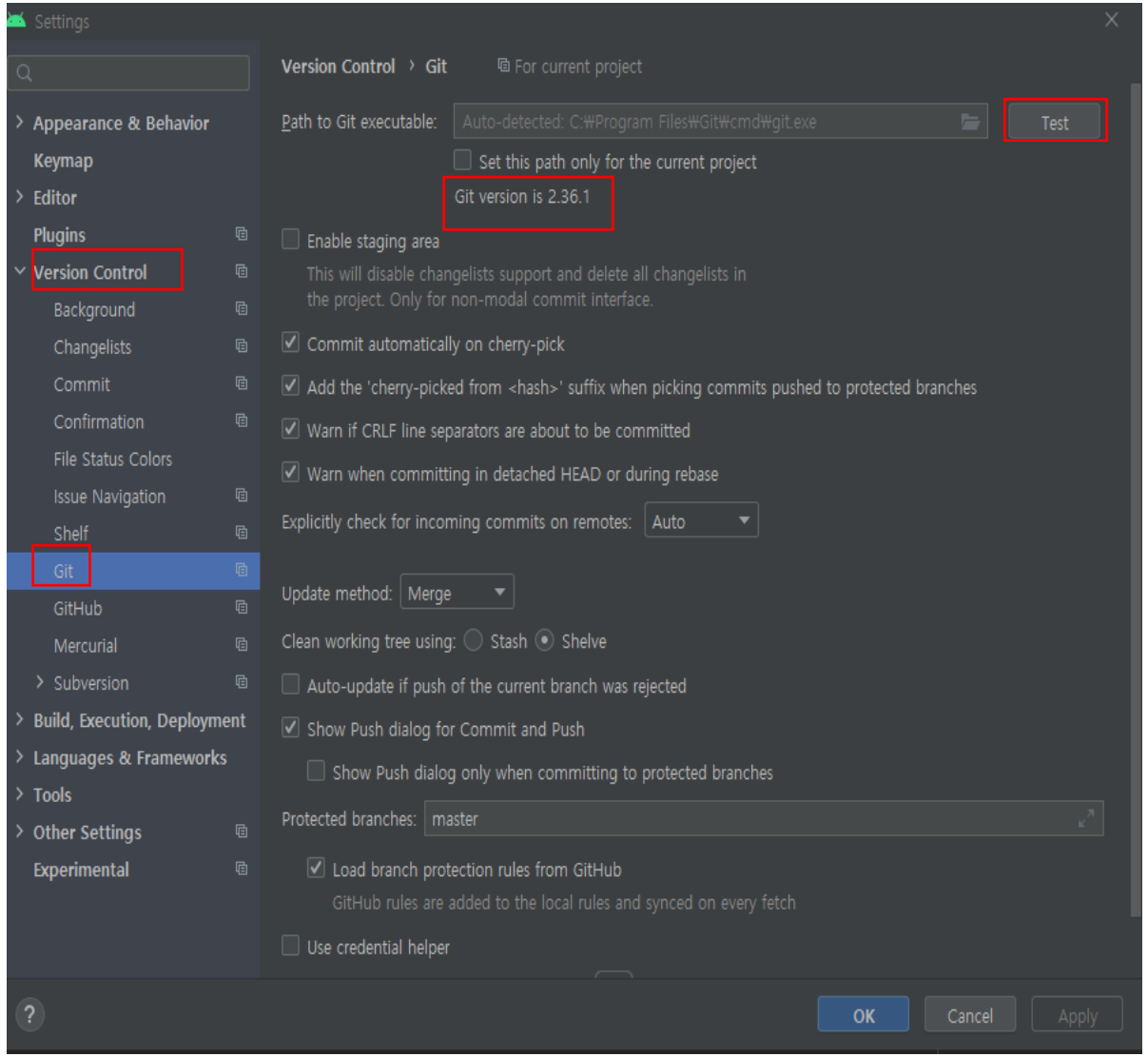
깃허브의 서비스들

- 저장소(Repository)
  핵심 기능은 깃(Git) 원격 저장소 호스팅
  로컬 개발 환경과 온라인에서 안정하게 깃 저장소에 접근

- 깃허브 페이지(GitHub Pages)-정적 웹 사이트 호스팅 서비스
  깃허브 저장소를 기반으로 정적 파일들을 호스팅할 수 있는 서비스
- 지속적인 통한 서비스인 깃허브 액션(GitHub Action)
- 패키지 저장소인 깃허브 패키지(GitHub Package)
- 깃허브 컨테이너 레시스토리(GitHub Container Registry)
- 깃허브 마켓플레이스(GitHub Marketplace)
- 개발자 정기 후원 서비스 등

5. 깃허브 연동

깃 버전 확인



깃(Git) 버전 확인

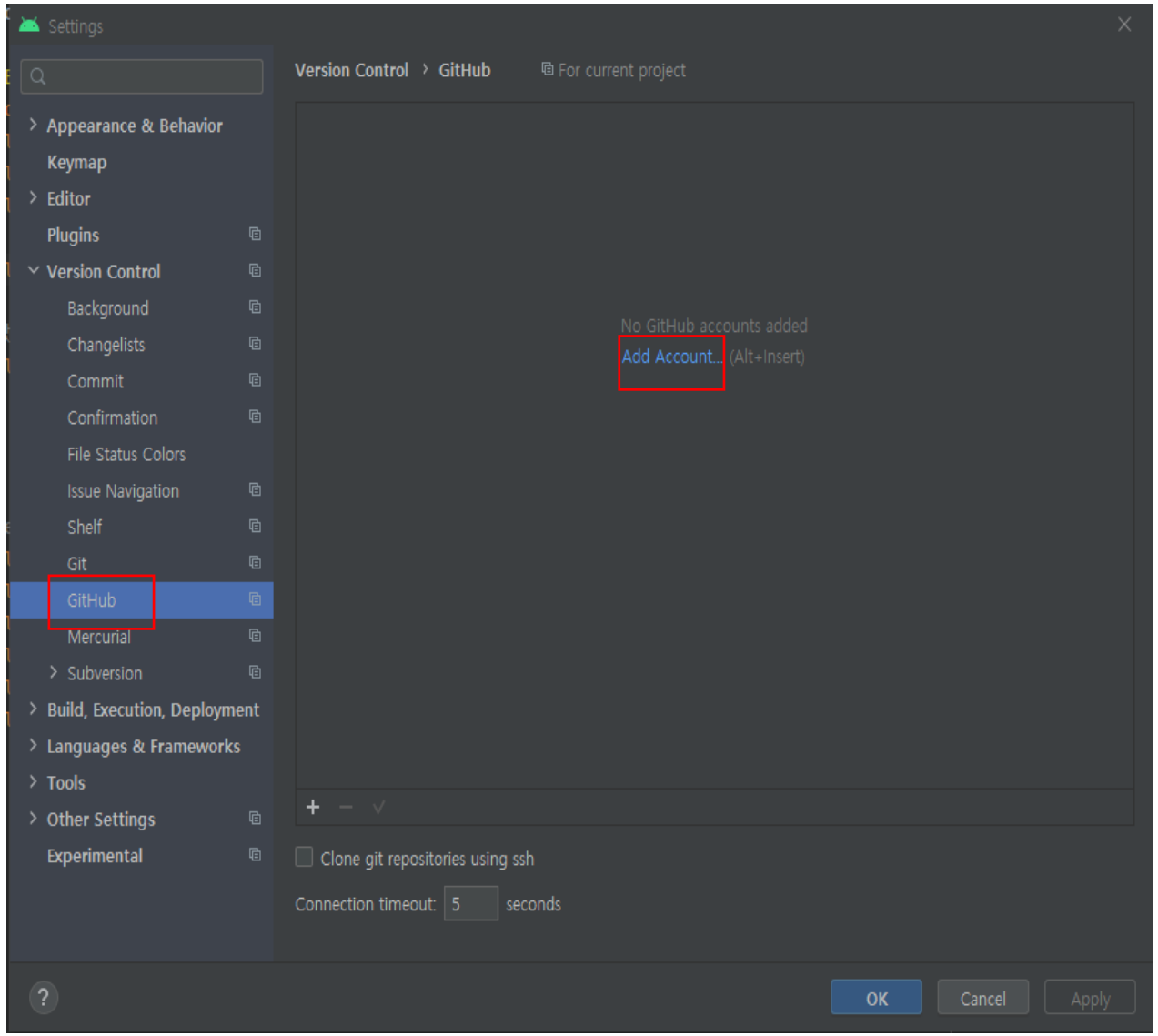
상단의 File -> Settings -> Version Control -> Git -> Test

Test버튼을 눌러 현재 설치된 깃의 버전을 파악

깃(Git)이 설치 되어 있어야 함

5. 깃허브 연동

안드로이드 스튜디오에 깃허브 연동하기

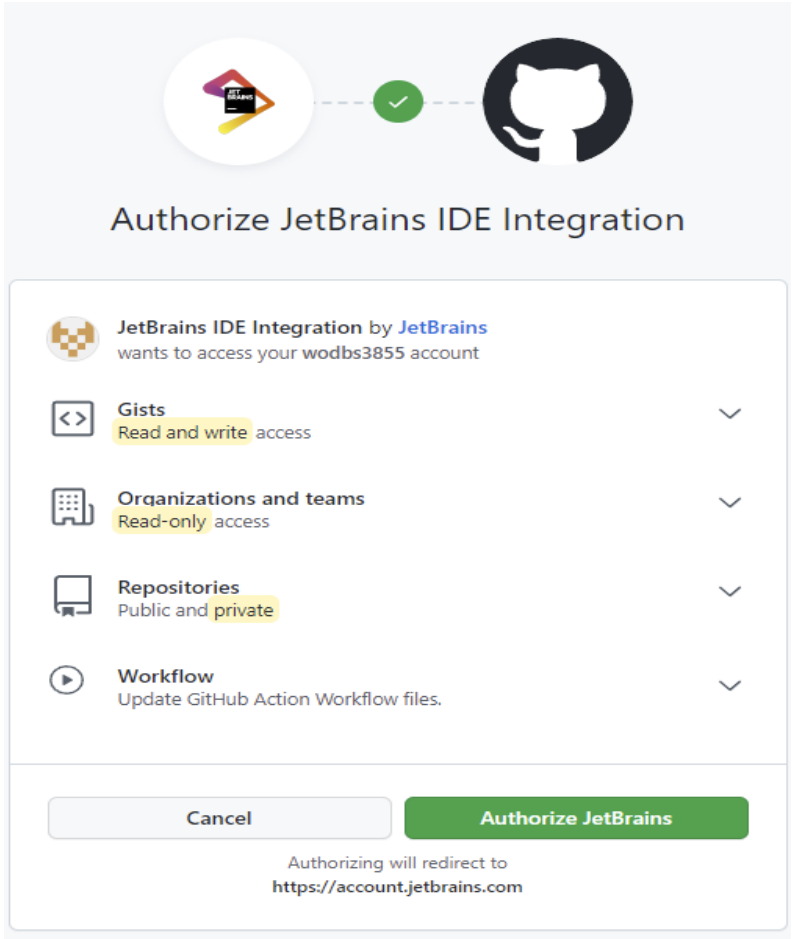


깃허브 연동

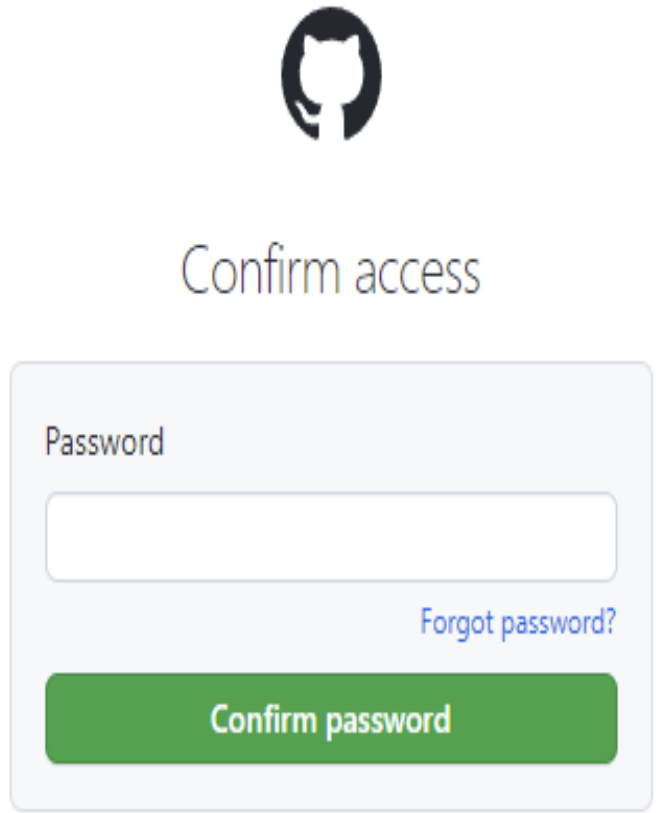
GitHub에서 add Account 클릭

5. 깃허브 연동

깃허브와 JetBrains통합



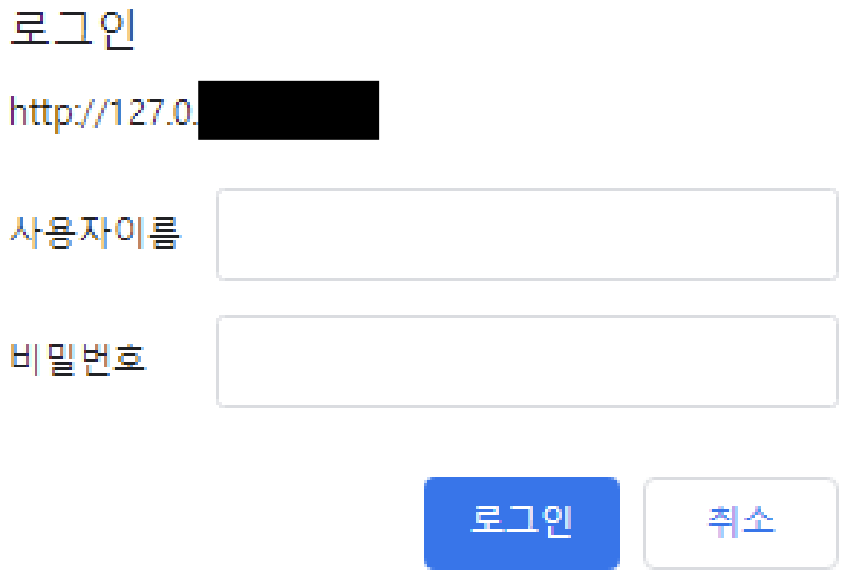
Authorize JetBrains클릭하여 GitHub 계정을 통합



비밀번호를 입력한 후에 **Confirm password 버튼**을 클릭

5. 깃허브 연동

안드로이드 스튜디오에 깃허브 계정 로그인

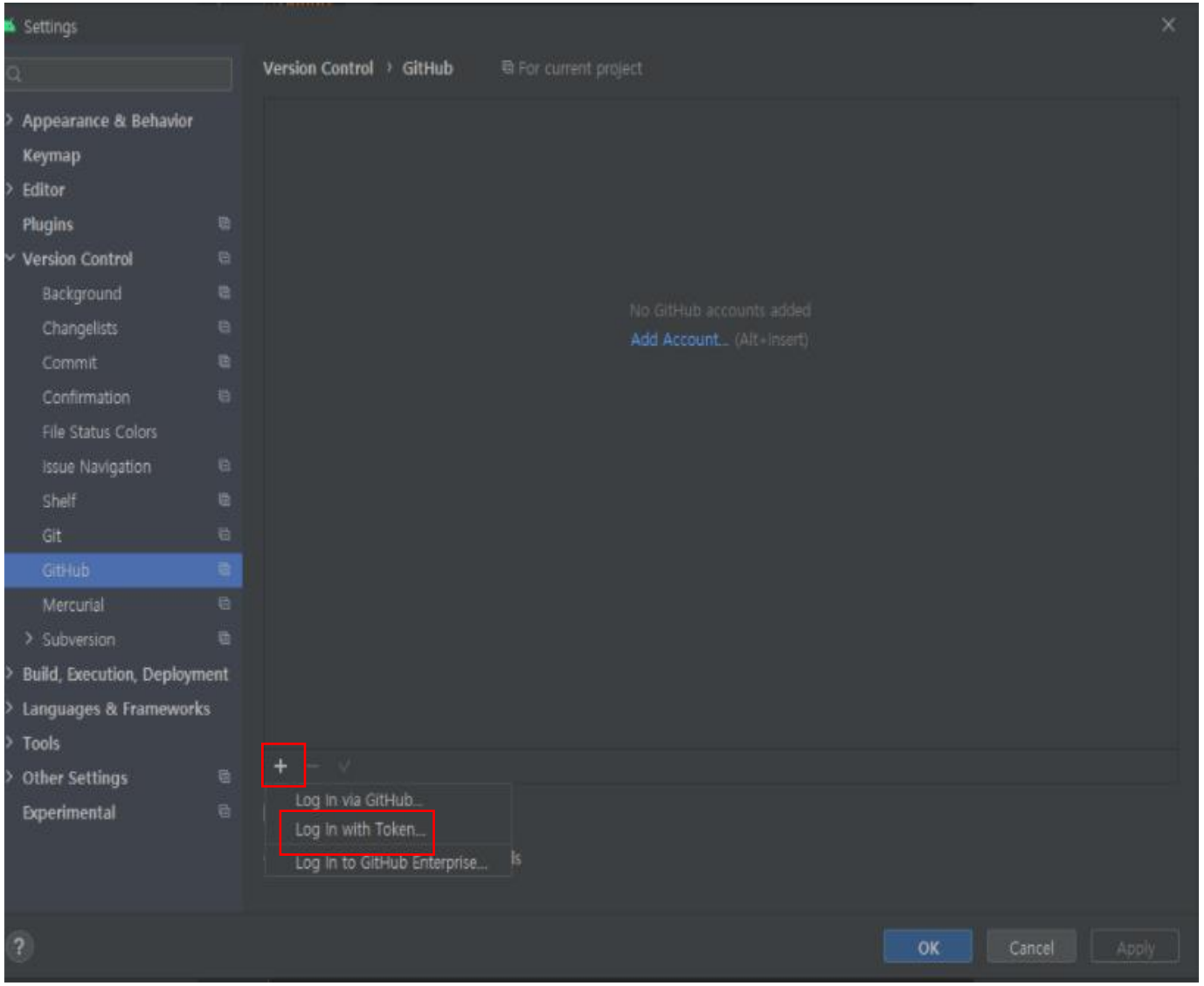


로그인 오류

GitHub 계정을 입력한 후 **로그인 버튼**을 클릭
로그인이 되면 문제가 없지만 로그인이 계속 안되는 오류 발생

5. 깃허브 연동

Token을 이용한 로그인



다른 방법으로 로그인

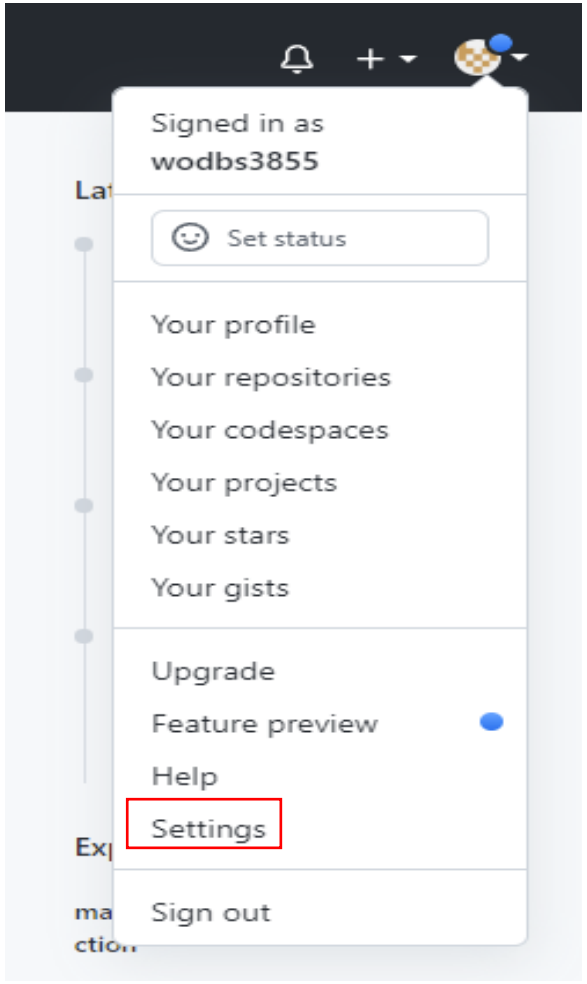
+버튼을 누르고 Log in with Token…을 클릭

Token을 이용한 로그인 방법

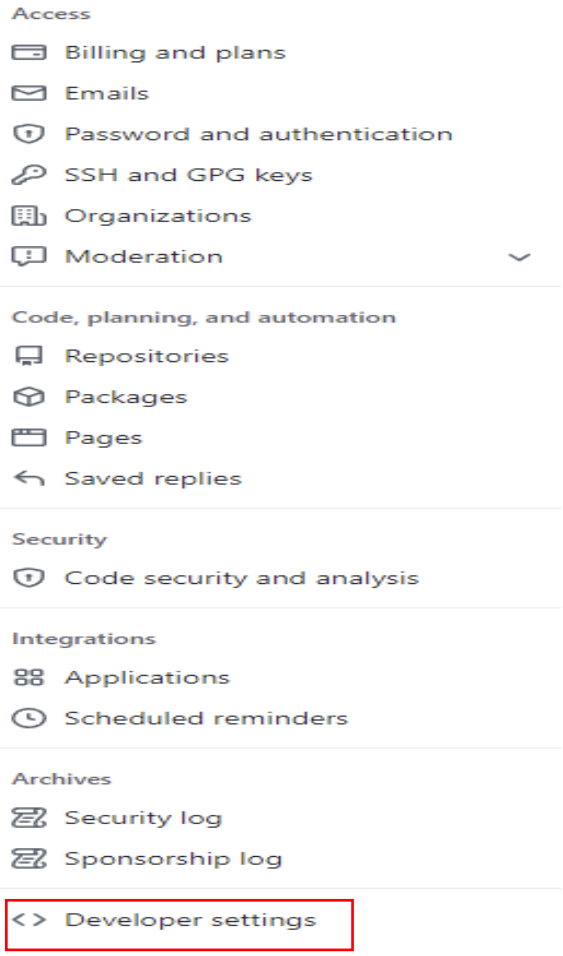
토큰이란?
일련의 문자열을 구분할 수 있는 단위 또는 시스템에서 보안 객체의 접근 관리에 사용되는 객체 또는 장치를 의미

5. 깃허브 연동

Token을 이용한 로그인



깃허브에서 Settings 클릭



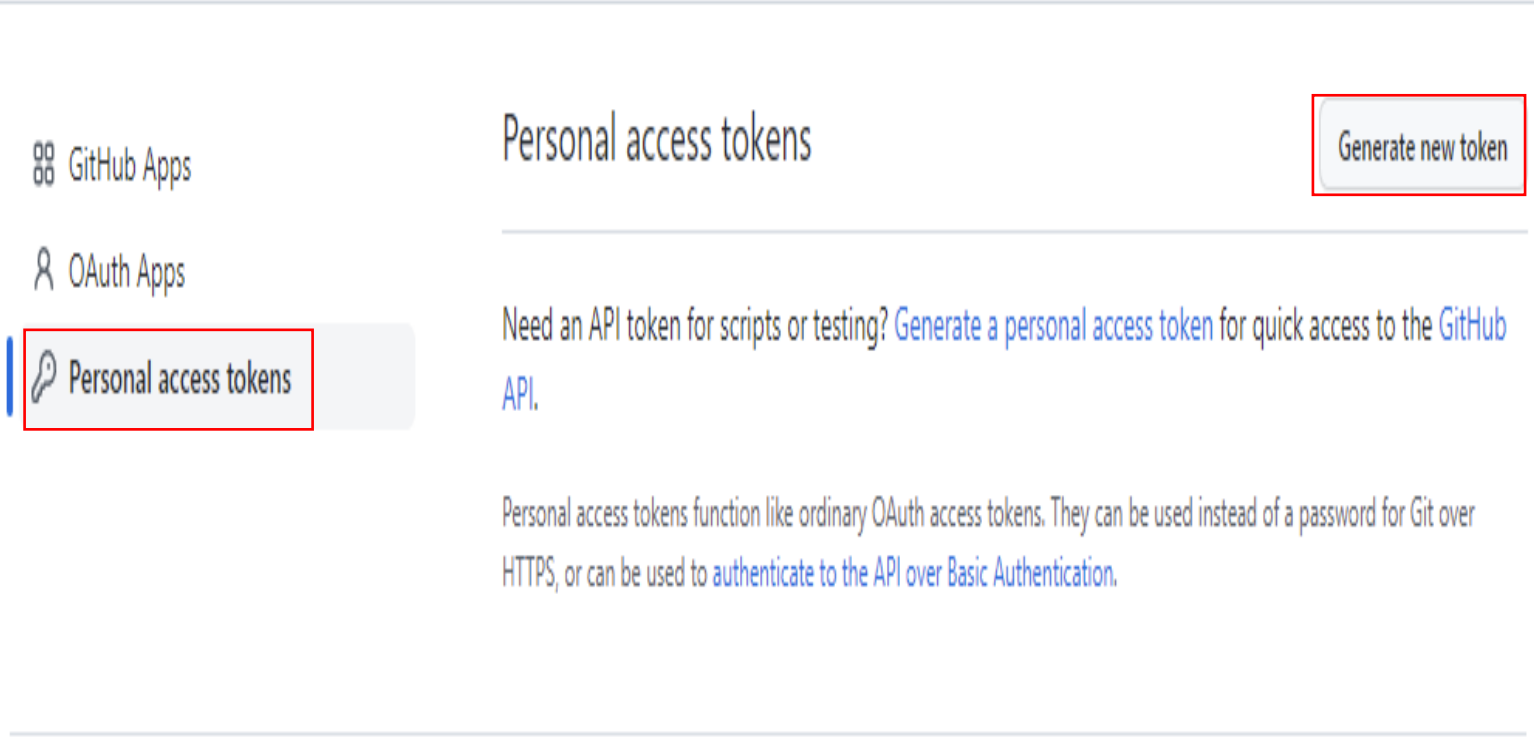
좌측 하단의Developer Settings클릭

5. 깃허브 연동

깃허브 토큰 생성

Token 생성하기

Personal access tokens클릭
Create new token클릭

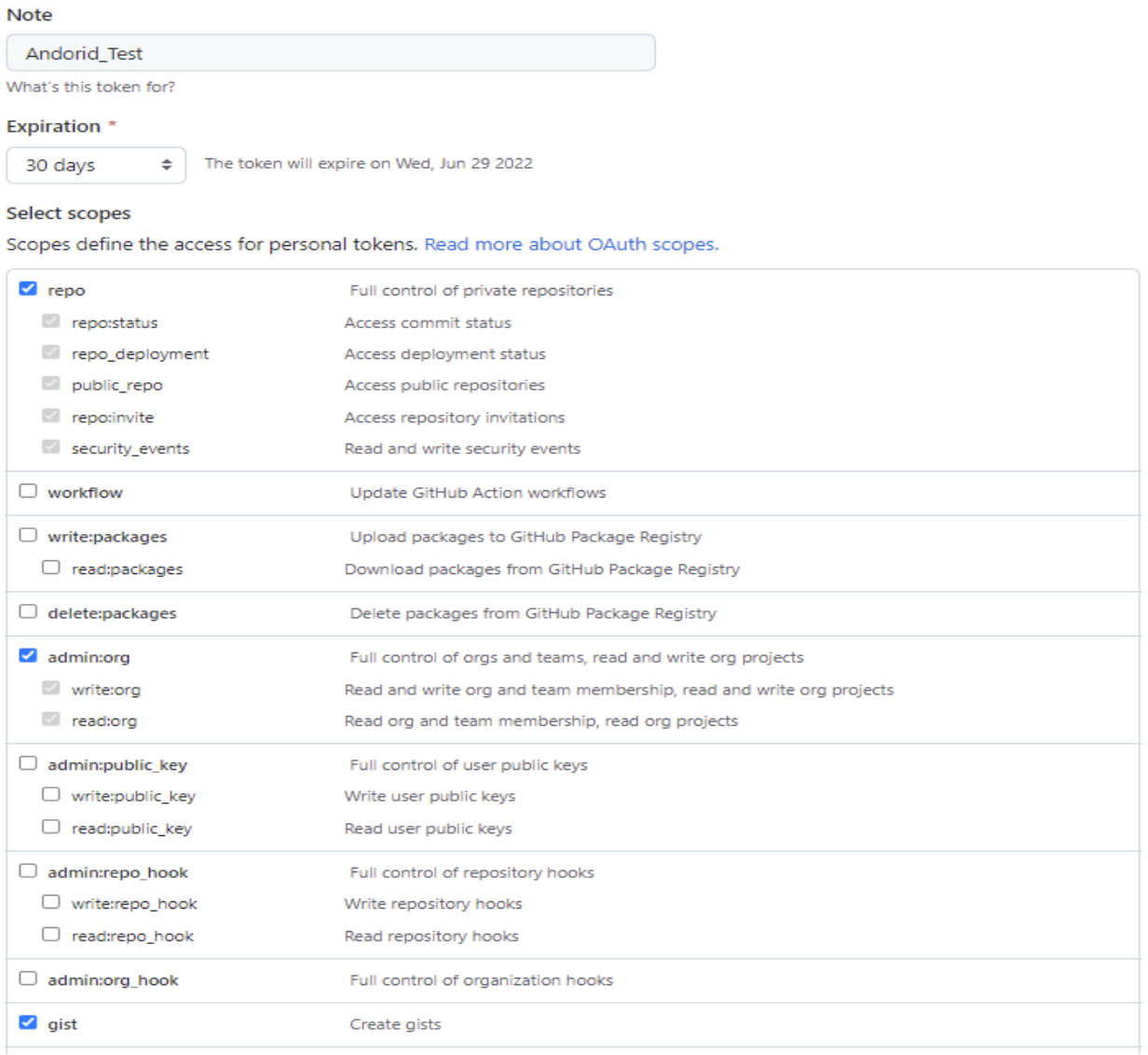


5. 깃허브 연동

깃허브 토큰 생성

Token 생성하기

repo, admin:org , gist를 체크하고 create 클릭

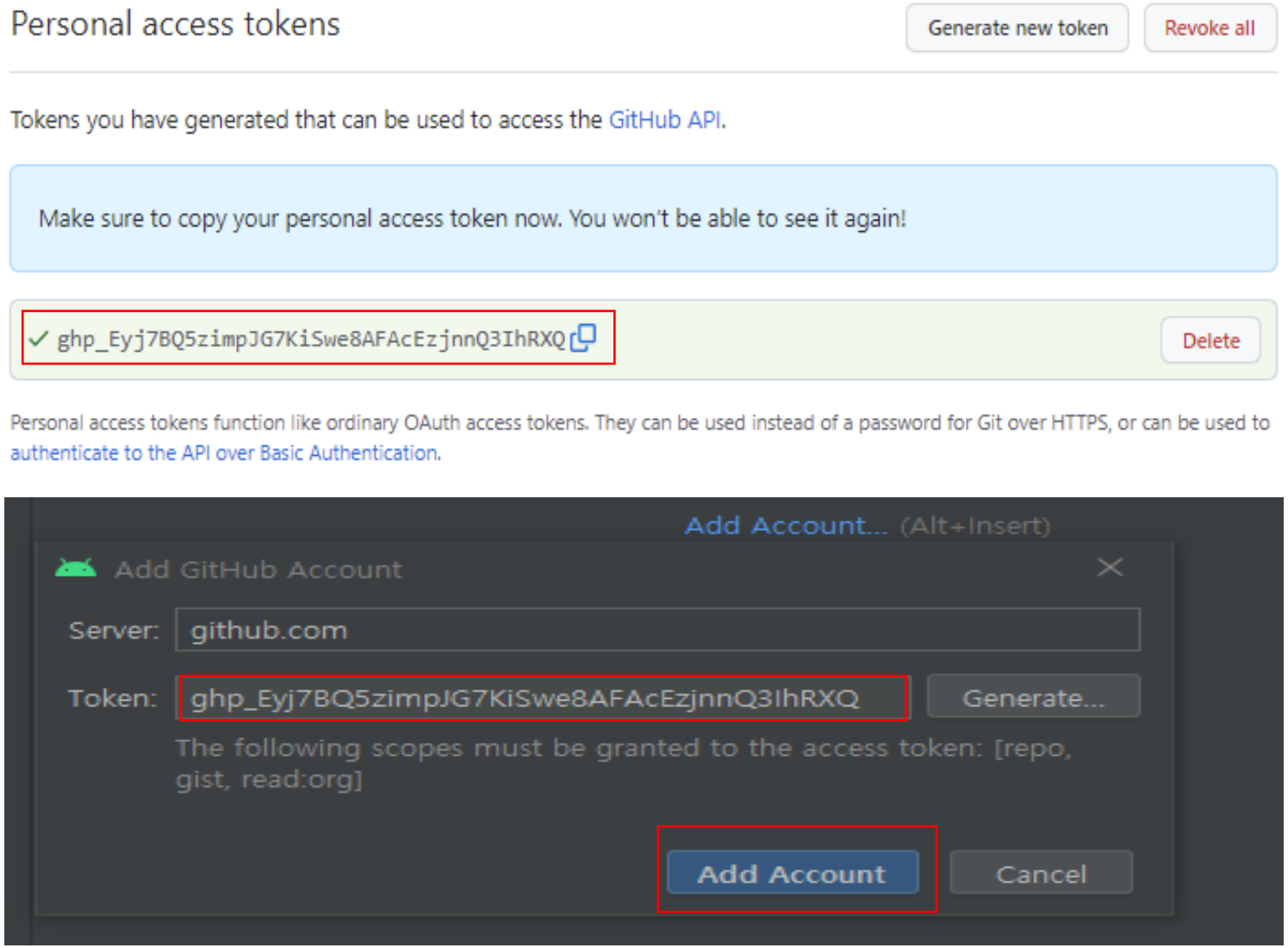


5. 깃허브 연동

깃허브 토큰 생성

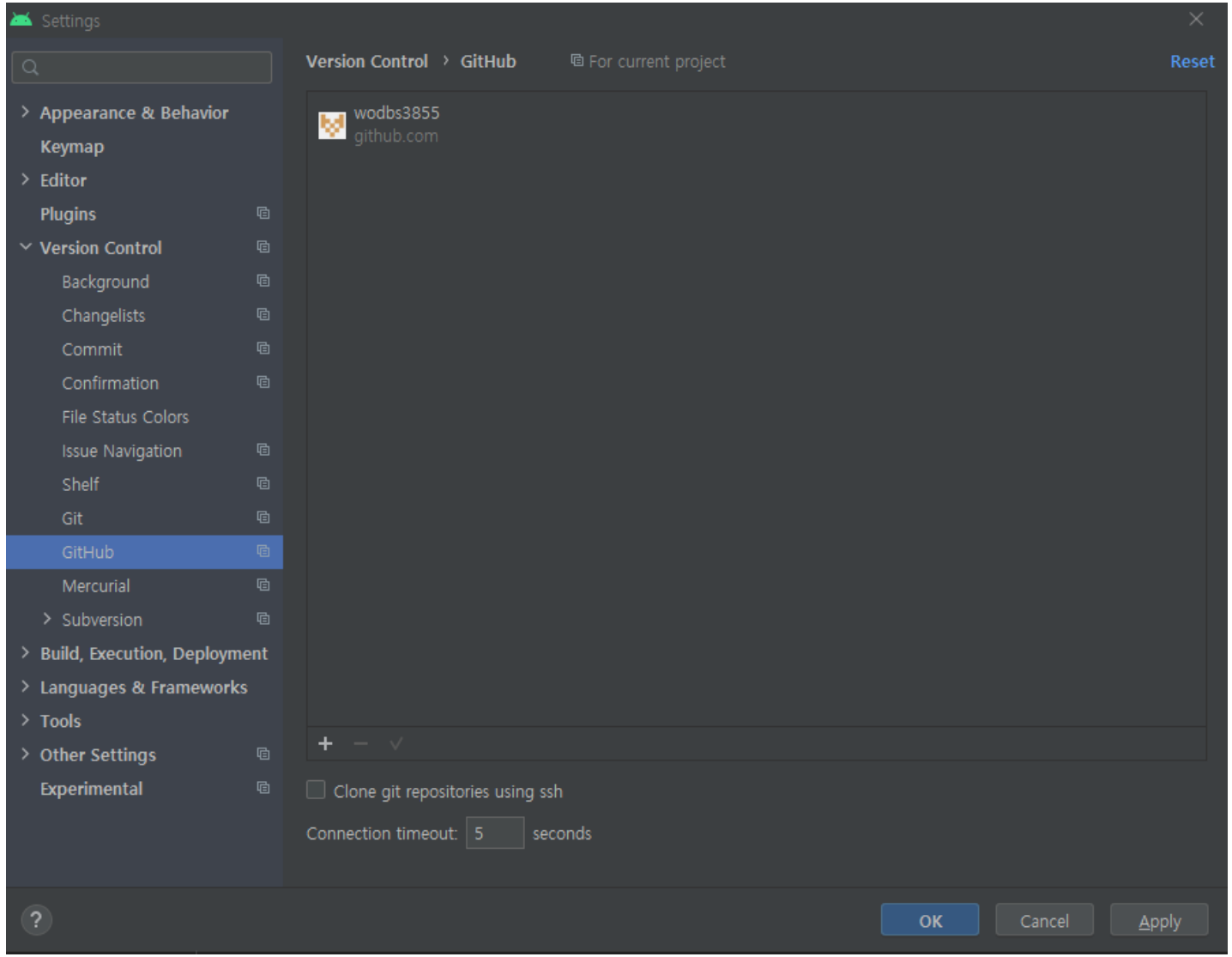
Token 생성완료

토큰 생성완료
생성된 토큰을 복사하여 붙여 넣고 Add Account 클릭



5. 깃허브 연동

안드로이드 스튜디오에 깃허브 계정 등록

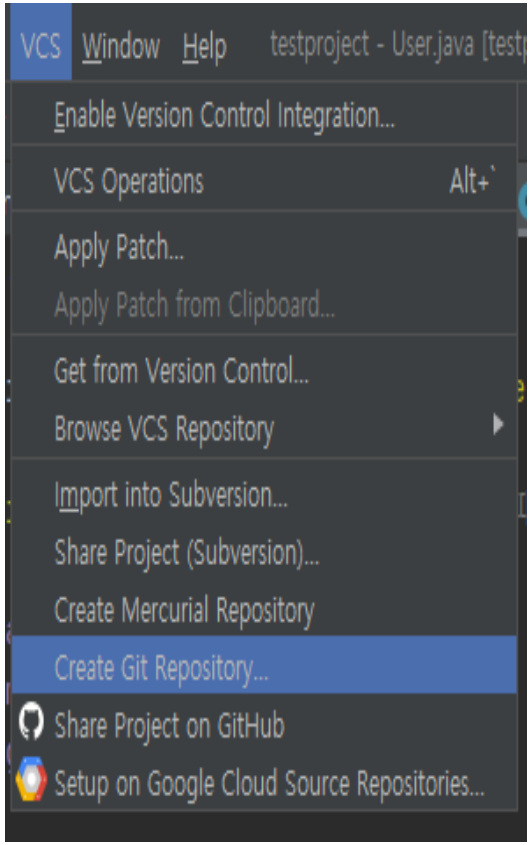


계정 등록 성공

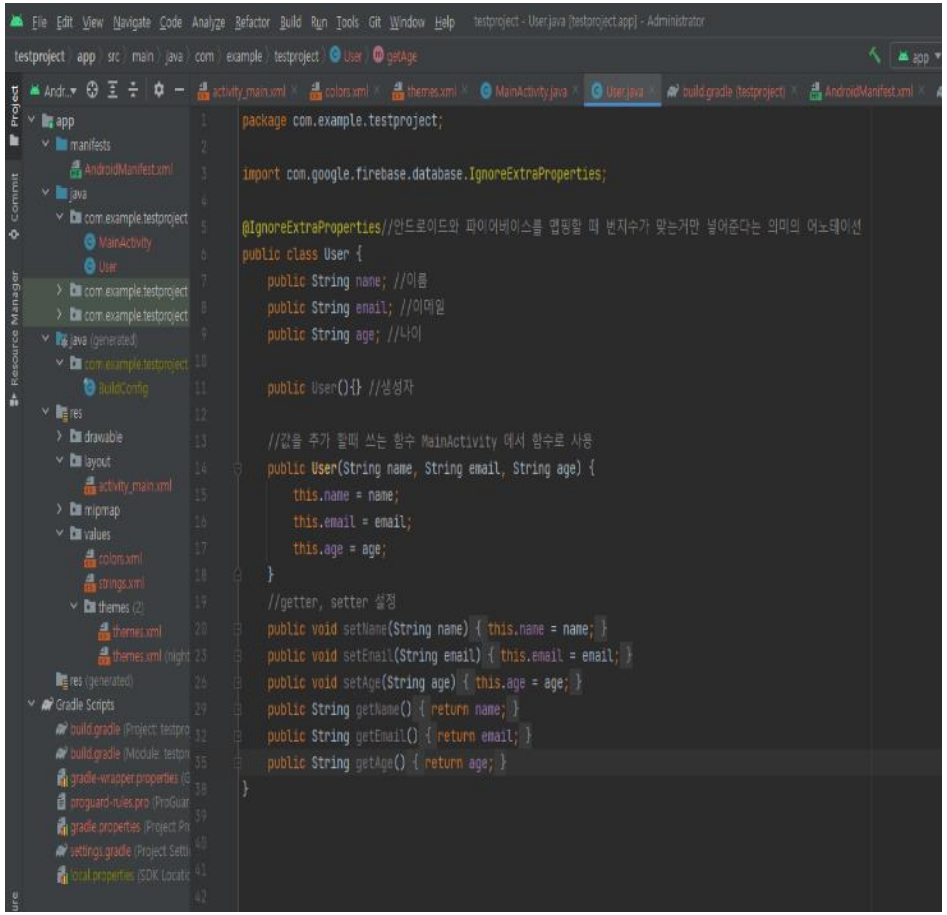
성공적으로 계정이 추가

5. 깃허브 연동

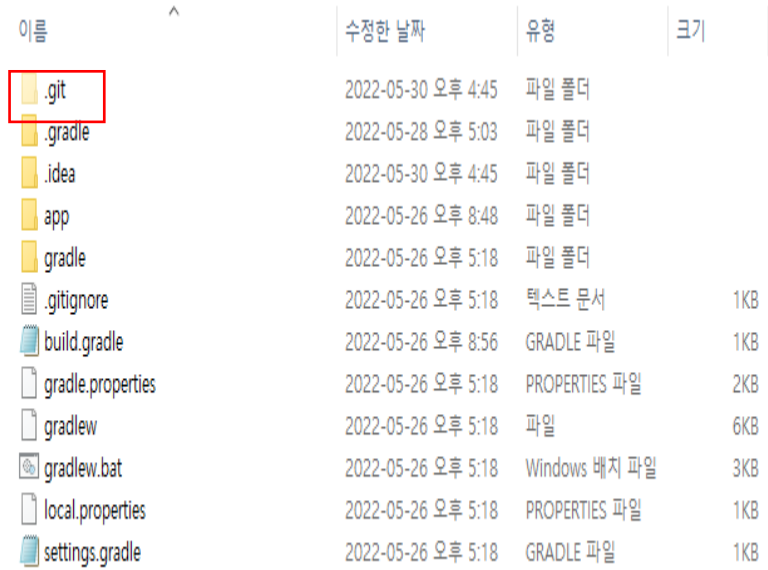
깃 저장소 생성



VCS에서 Create Git Repository클릭



파일들이 빨간색으로 표시 -> 깃허브에 올라가지 않은 새로운/수정된 코드

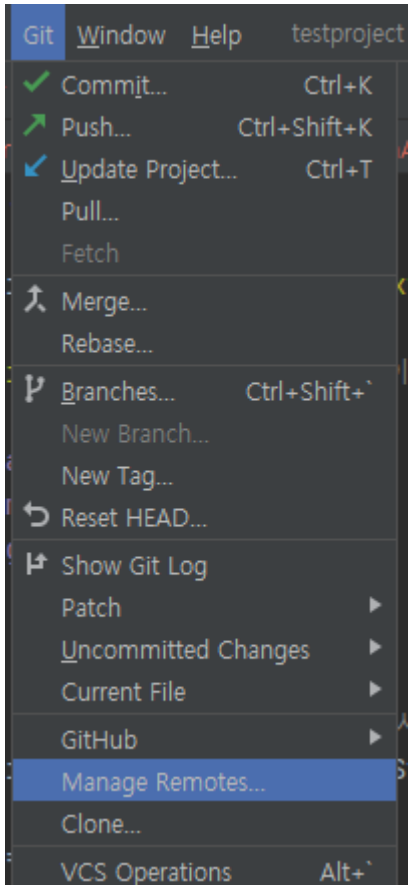


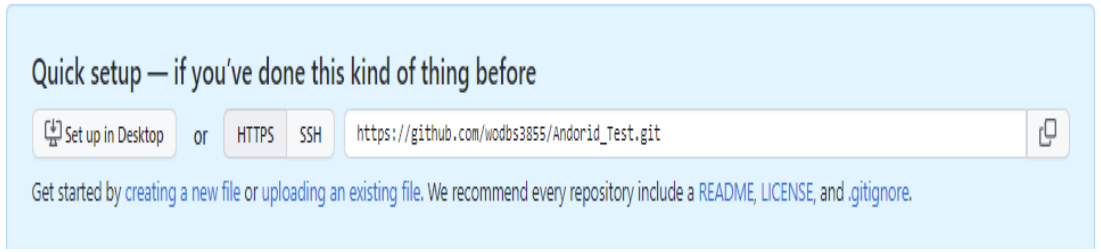
이름	수정한 날짜	유형	크기
.git	2022-05-30 오후 4:45	파일 폴더	
.gradle	2022-05-28 오후 5:03	파일 폴더	
.idea	2022-05-30 오후 4:45	파일 폴더	
app	2022-05-26 오후 8:48	파일 폴더	
gradle	2022-05-26 오후 5:18	파일 폴더	
.gitignore	2022-05-26 오후 5:18	텍스트 문서	1KB
build.gradle	2022-05-26 오후 8:56	GRADLE 파일	1KB
gradle.properties	2022-05-26 오후 5:18	PROPERTIES 파일	2KB
gradlew	2022-05-26 오후 5:18	파일	6KB
gradlew.bat	2022-05-26 오후 5:18	Windows 배치 파일	3KB
local.properties	2022-05-26 오후 5:18	PROPERTIES 파일	1KB
settings.gradle	2022-05-26 오후 5:18	GRADLE 파일	1KB

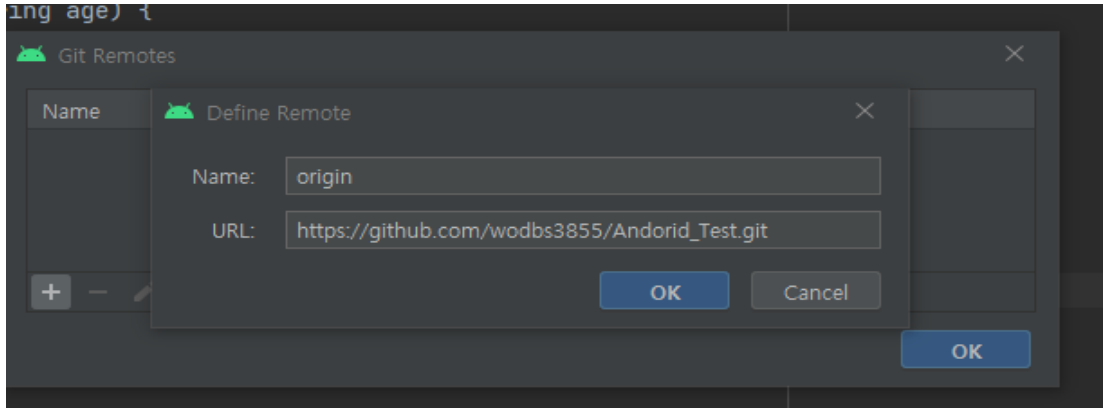
폴더에 .git파일이 추가된 것을 확인

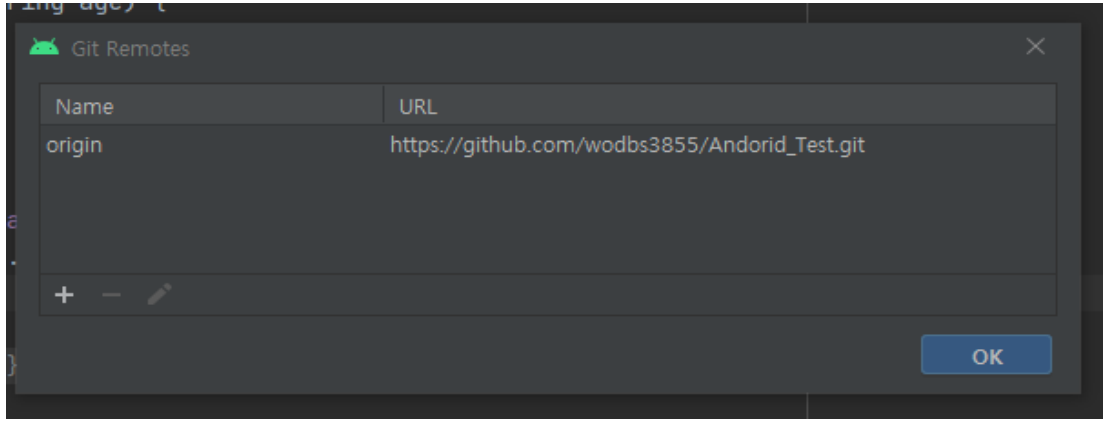
5. 깃허브 연동

깃허브와 안드로이드 스튜디오 연동









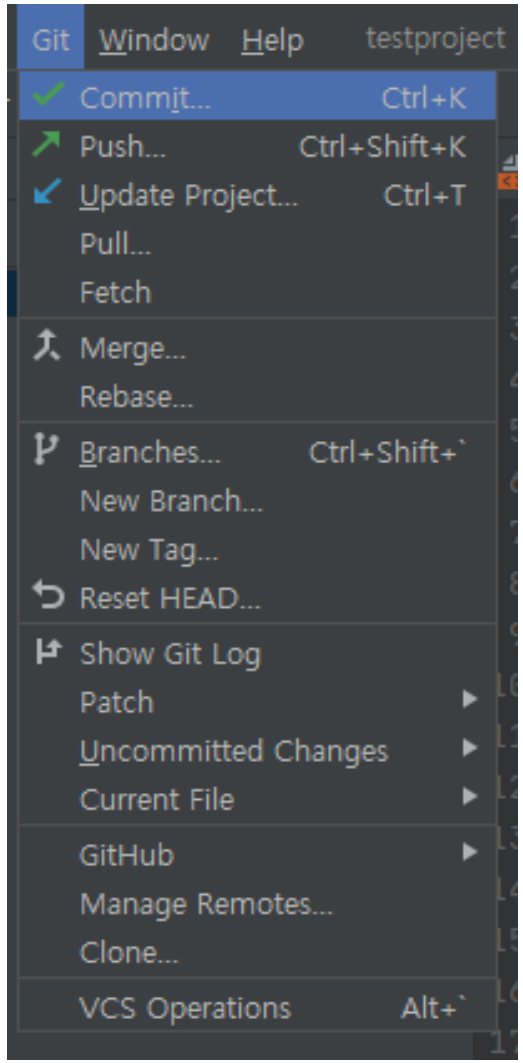
깃허브의 저장소와 연동

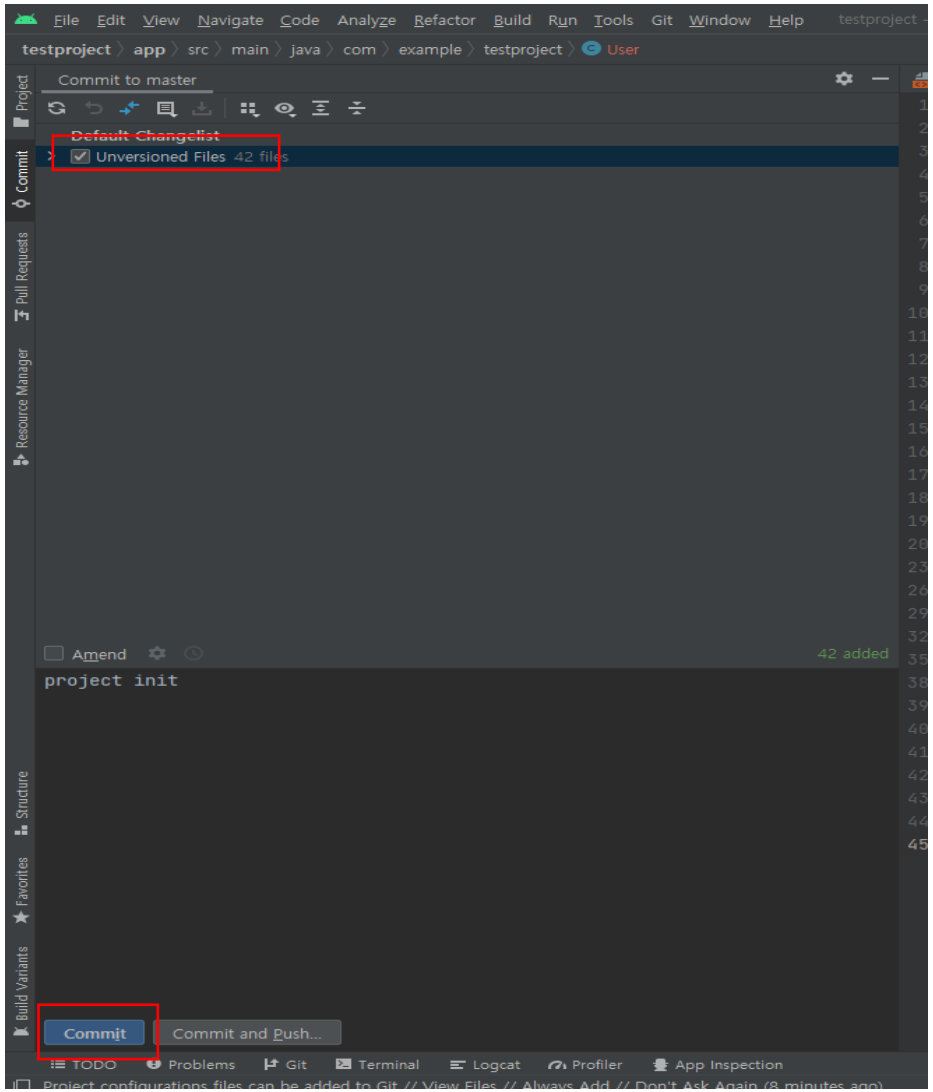
Git의 Manage Remotes클릭

새로운 저장소를 만들거나 기존의 저장소의 HTTPS의 주소의 경로를 붙여 줌

5. 깃허브 연동

Commit하기





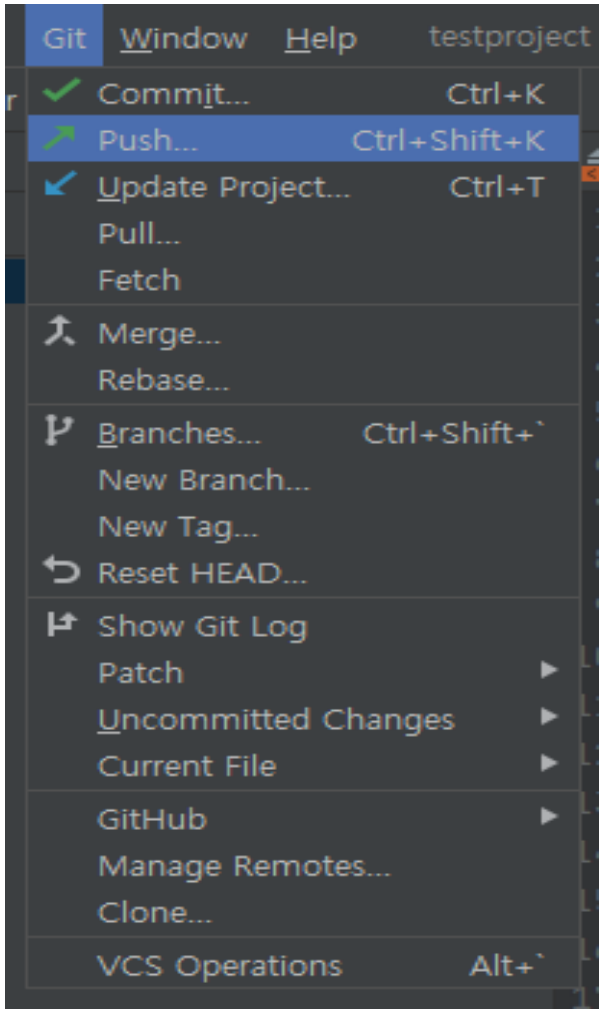
Commit하기

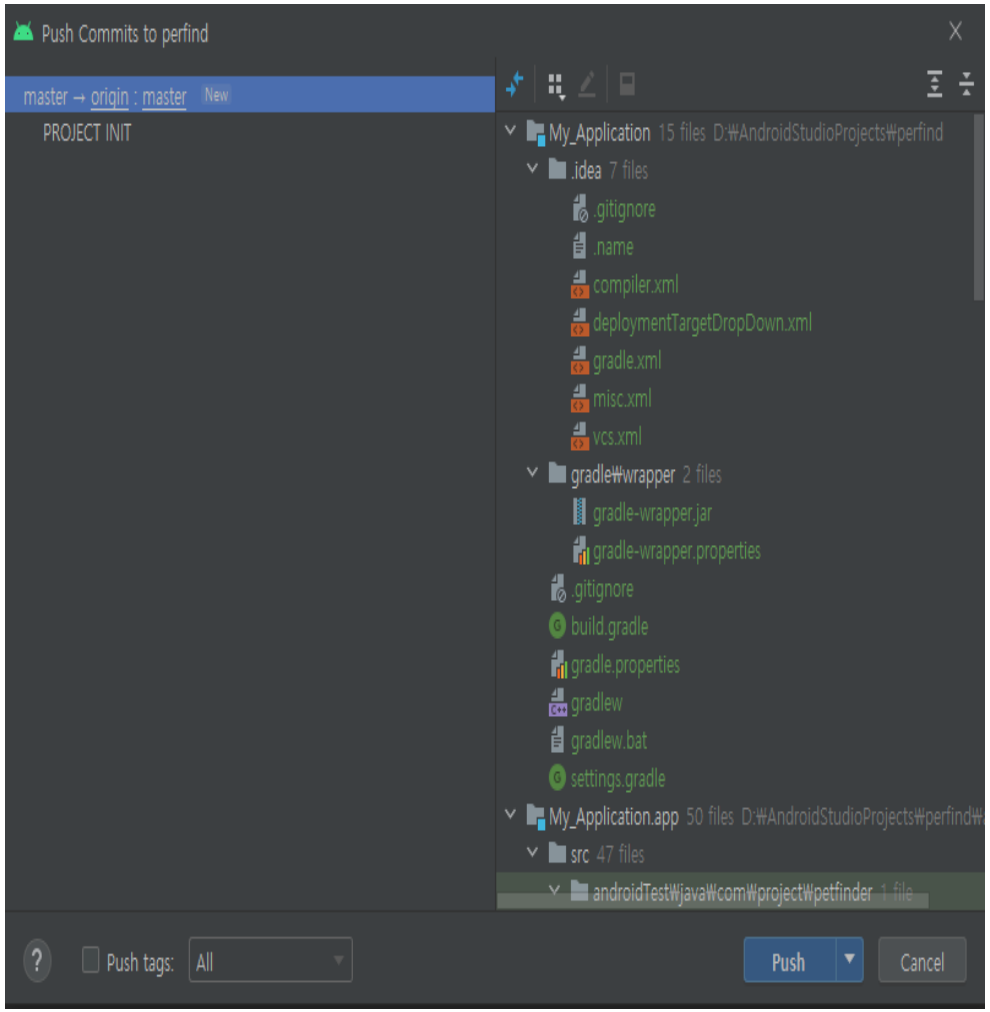
Git에서 Commit클릭
Commit할 파일들 체크
하단의 Commit을 클릭

Commit이란?
파일 및 폴더의 추가/변경 사항을 저장소에 기록하는 것.

5. 깃허브 연동

깃허브에 Push하기





Push하기

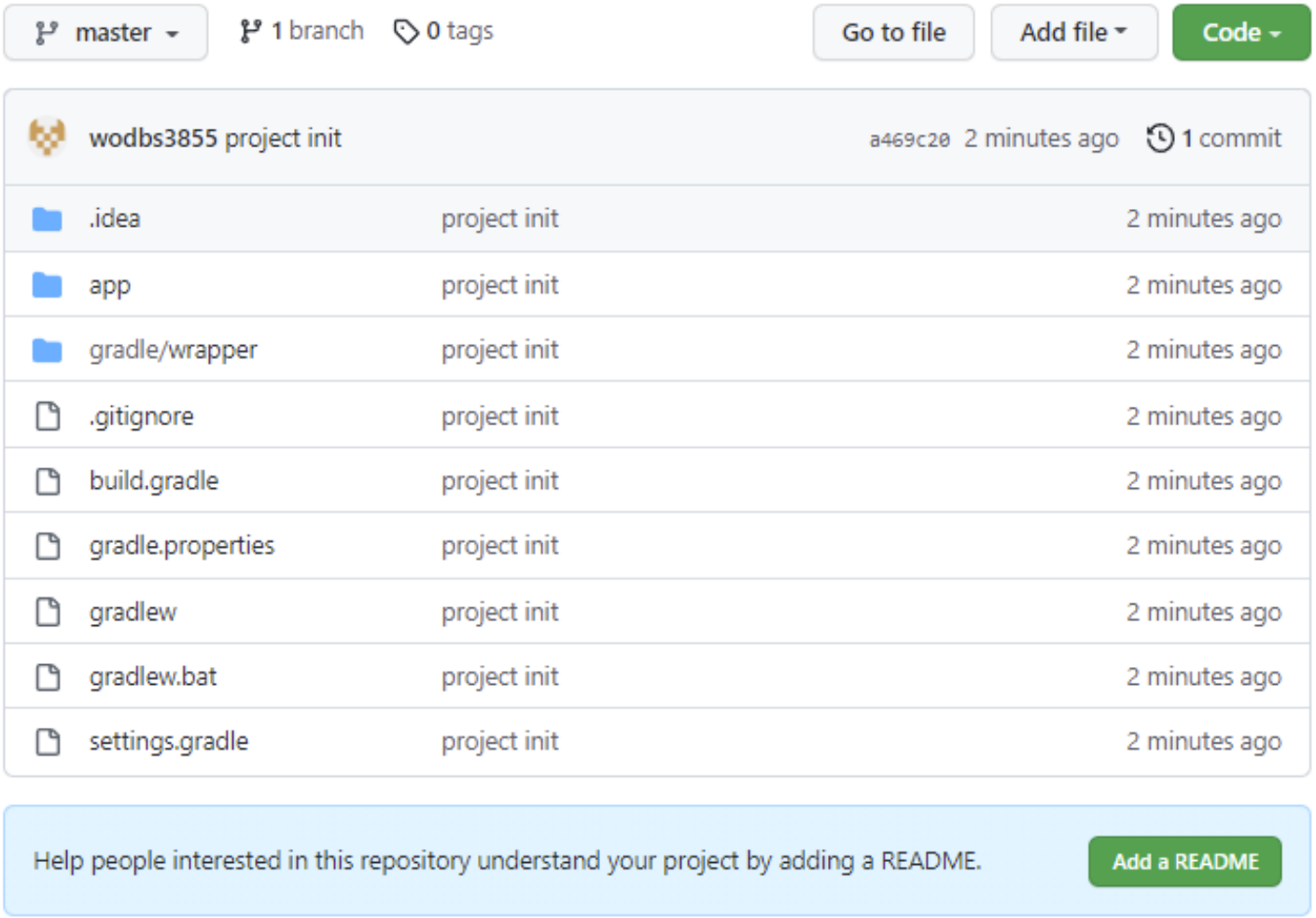
Git의 Push를 클릭

Commit했던 파일들을 확인할 수 있음
초록색 파일들이 깃허브에 업로드할 파일

Commit한 후 반드시 Push를 눌러줘야
GitHub의 저장소에 올라감

5. 깃허브 연동

깃허브 저장소 확인



GitHub 저장소 확인

선택한 파일들이 저장소에 업로드가 된 것을 확인

5. 깃허브 연동

깃허브 저장소 확인

수정내용(변경사항)을 협업 프로젝트에 최신으로 업데이트

Update Project도구를 눌러주면 자동으로 수정된 사항만 업데이트 됨

상단메뉴 VCS -> Git -> Pull을 통해서도 가능

(선택적으로 업데이트 되는 것이 아니라 모든 사항이 업데이트 됨에 주의

감사합니다