

Firestore Cloud Firestore for Android App

안드로이드 앱을 위한 파이어베이스 파이어스토어

<https://github.com/2tle>







Cloud Firestore?

Cloud Firestore란?

NoSQL 클라우드 데이터베이스

- SQL 데이터베이스와 달리 테이블이나 행 없음
- 컬렉션으로 정리되는 문서에 데이터 저장
- 각 문서는 키-값 쌍으로 저장됨

Cloud Firestore 형태

 today-kotlin	 users  	 0WW5m2X7jVQx2YgiDRhAZAnuUtp2 
+ 컬렉션 시작	+ 문서 추가	+ 컬렉션 시작
luckyItems	0WW5m2X7jVQx2YgiDRhAZAnuUtp2 >	+ 필드 추가
posts	5tSNk98Y2uVJnFRoPhg7iNWdt8N2	date: "2021-07-14"
users >	6p8gFYxA7Hc3UH1WWG0I0bQp4Js1	item: "당신의 휴대폰화"
words	DM9tt7JJTyaf9wZfuBfIiqpTwkT2	itemSrc: "https://firebasestorage.googleapis.com/v0/b/today-kotlin.appspot.com/o/items%2Fcellphone.png?alt=media&token=eada8758-b2ee-44cc-ade3-0ee90acdd9be"
	GhIZZL1WP2NFkwQjkhztoZoHLCj2	▼ listDates
	I9wB11Cz2zY0Kc7MGGQ0AUIKzvn2	0 "2021-07-14"
	Ja0y8aM0LwgDbGARwp7NYaTjTZF2	▼ listWords
	K9gx4wK60VeDrz9jYrojitKSsj83	0 "주변사람이 더 잘 알거다"
	N5jXAs4A5VM6QUV0oYHuz0x00dG3	todayWords: "주변사람이 더 잘 알거다"
	NUwpBF6BibZektXzXF23lp1WIpf2	
	Qv2FcTOML7NJDwGXE8jhqqN6YDD2	
	UTUX7SSpMufPa16AhN87MDGq9qj1	
	Wd2ZSf8FAnTyK1UuZ7D5pgX01sH2	
	XutbFIVmsu098EyHDdrEn0JwDfq1	

Cloud Firestore 형태

컬렉션 -> 비슷한 데이터(e.g. 유저 데이터)를 묶어놓은 것

문서 -> 각각의 유저 데이터를 의미

필드 -> 각각의 유저 데이터의 내용(e.g. 이름, 생일, 나이..)

Cloud Firestore가 지원하는 데이터 타입

문자열

숫자

불린

맵

배열

Null

타임스탬프(시간)

Geopoint(위도경도)

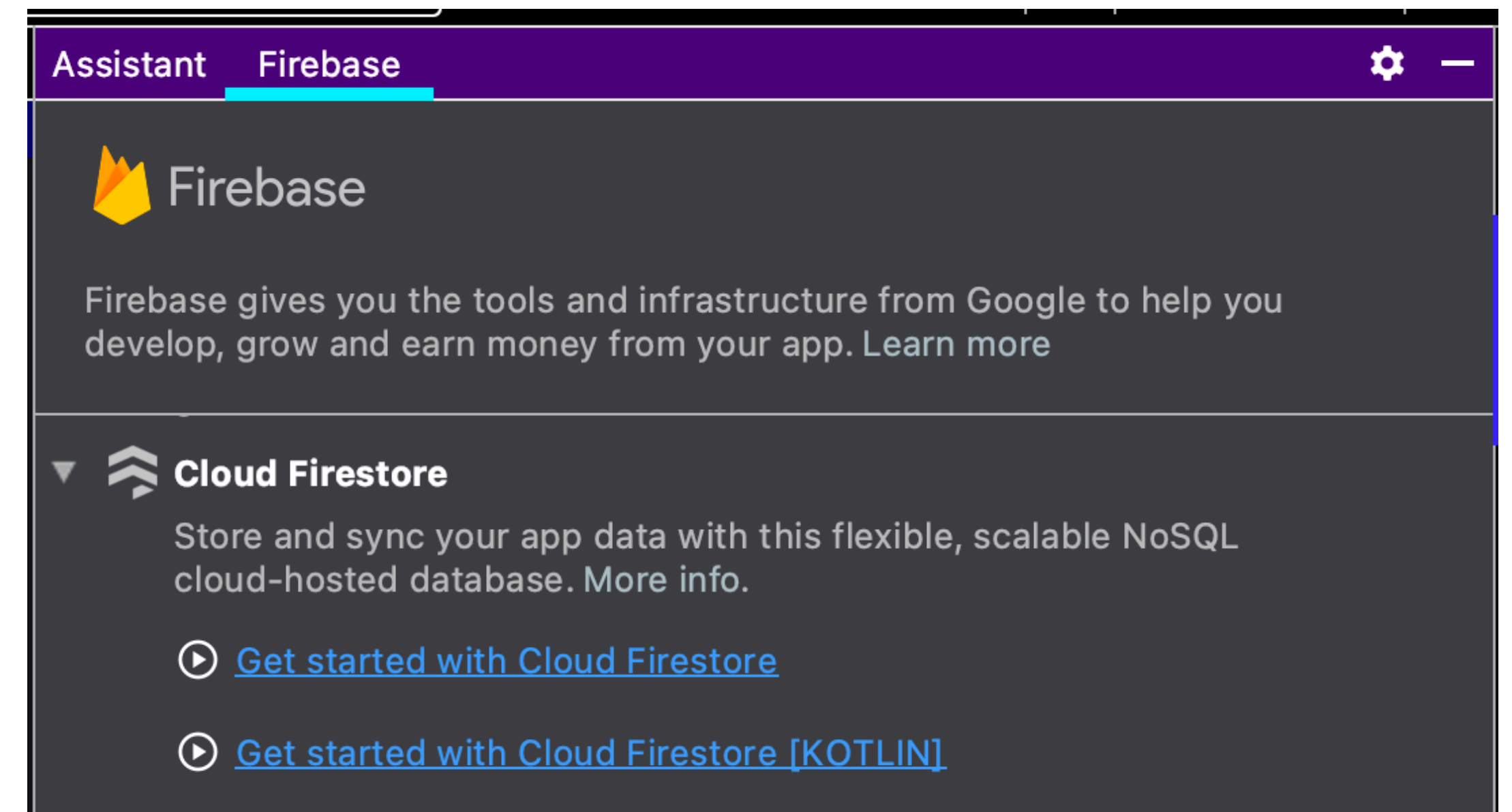
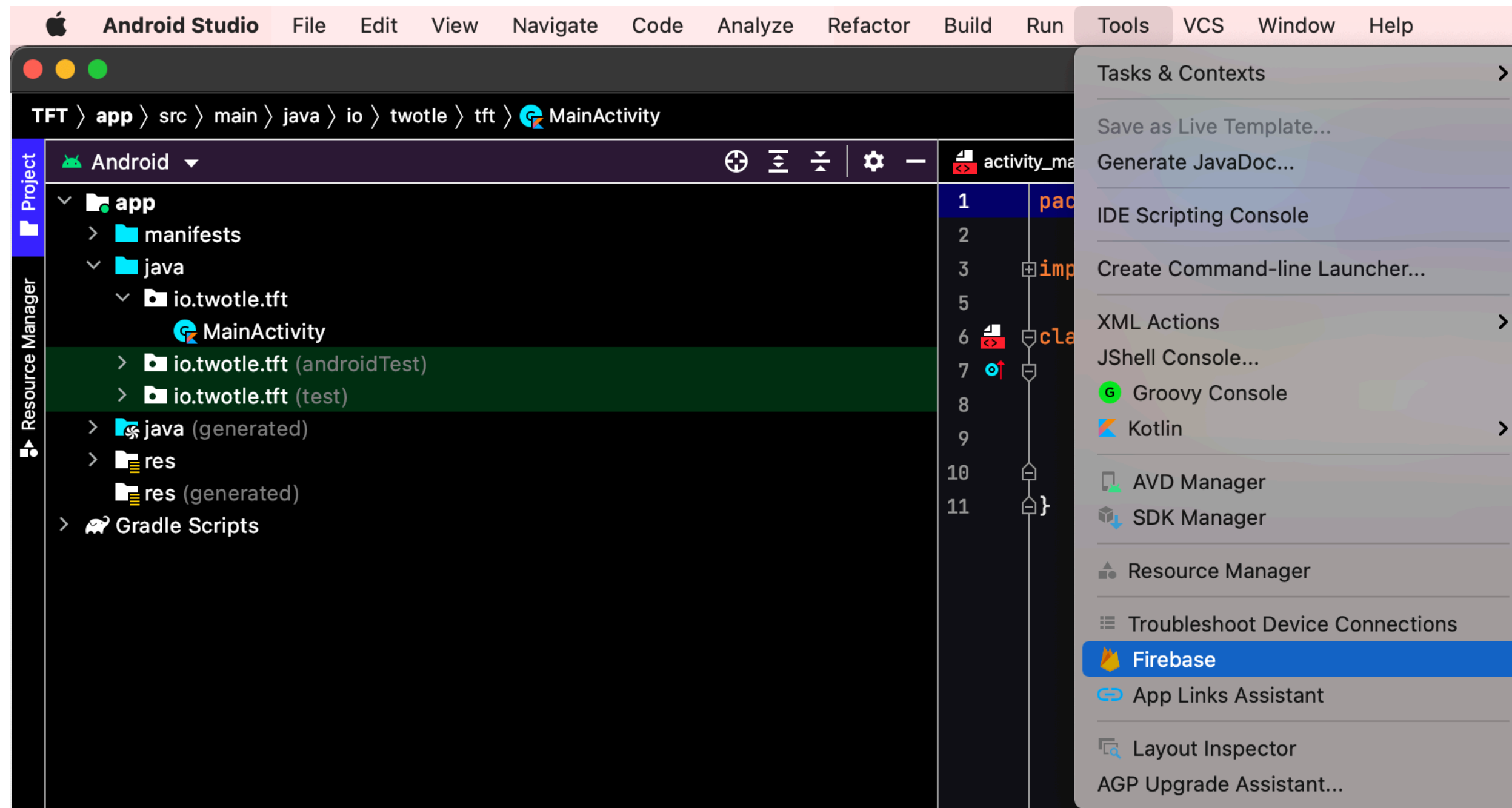
유형	값
=	string
	number
	boolean
	map
0	array
1	null
"	timestamp
	geopoint

Cloud Firestore를 프로젝트와 연결

안드로이드 스튜디오에서 Firestore 연결

Tools > Firebase

Cloud Firestore > Get started 클릭 (자신의 언어에 맞는거 선택)



안드로이드 스튜디오에서 Firestore 연결

Add the Cloud ... your app > Accept Changes

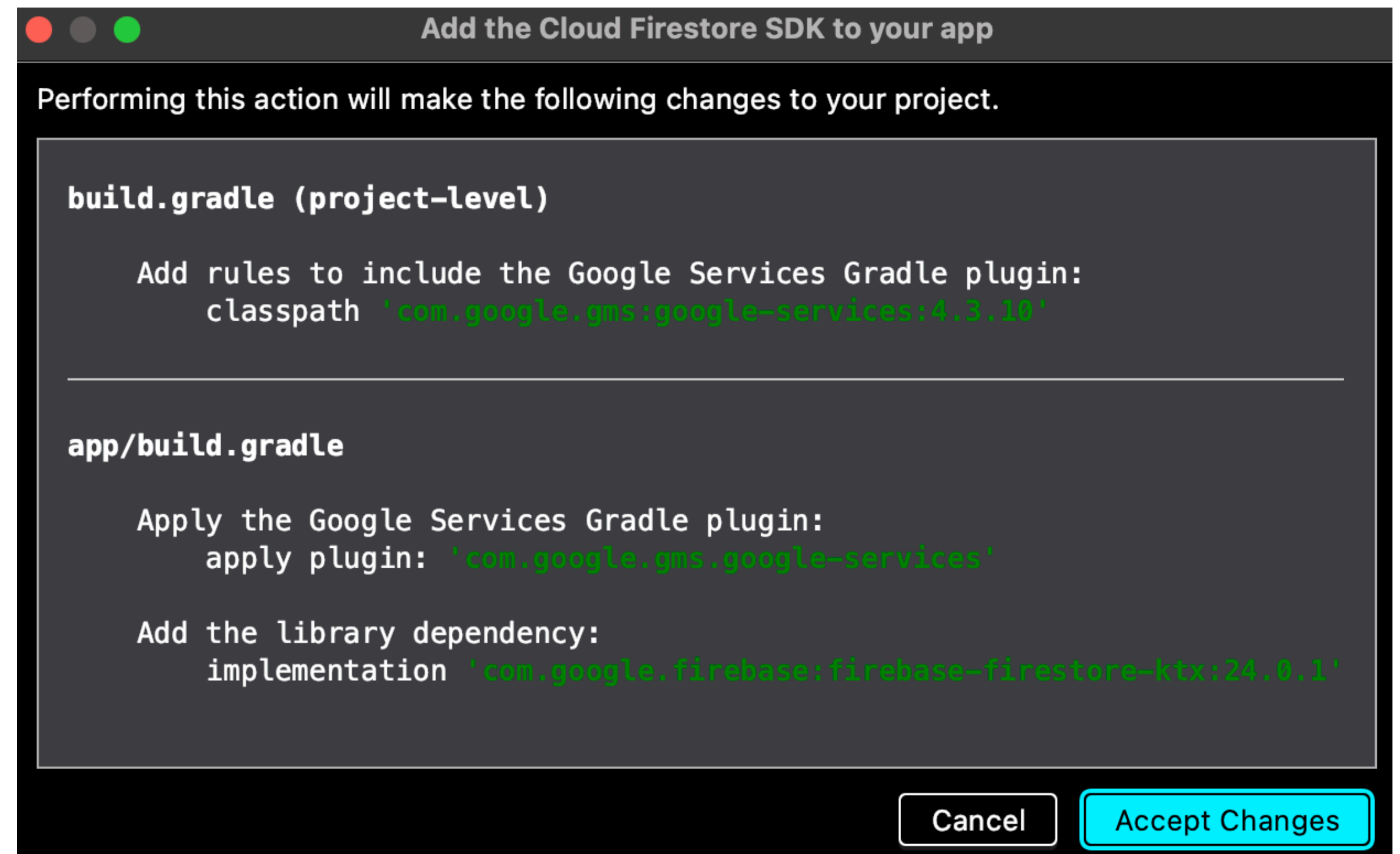
// Firebase Firestore 추가하는 과정

② Add Cloud Firestore to your app

Add the Cloud Firestore SDK to your app

② Add Cloud Firestore to your app

✓ Dependencies set up correctly



Cloud Firestore

Firestore 초기화

Firestore를 사용할 Activity.kt 에서
Firestore 인스턴스를 초기화 해 줍니다.



```
val db = Firebase.firestore
```

Firestore 데이터 추가

users 컬렉션에 저장될 문서를 추가해봅시다.

Firestore는 컬렉션이 존재하지 않아도 데이터 추가시 알아서 만들어줍니다.

*hashMapOf -> 키와 값을 가지는 형태의 데이터 컬렉션

Firestore 데이터 추가



```
val db = Firebase.firestore

val user = hashMapOf(
    "username" to "hyunju",
    "department" to "SW",
    "born" to 2005
)

db.collection("users")
    .add(user)
    .addOnSuccessListener { documentReference ->
        Log.d("Main>", "다음 아이디로 문서 생성: ${documentReference.id}")
    }.addOnFailureListener { e ->
        Log.w("Main>", "오류 발생:", e)
    }
```

Firestore 단일 문서 생성

단일 문서를 만들거나 덮어 씌우려면 `set()` 메서드를 사용함

- `set()` 사용시 문서가 없다면 생성, 있다면 덮어쓰기 됨
- 만약 기존 문서와 병합하고 싶다면 `set()` 메서드의 두번째 인자로 `SetOptions.merge()` 를 넘겨줘야함

* `set()` 메서드를 사용해 문서를 만들때에는 문서의 ID를 명시해야함.
다만, 문서에 유의미한 ID가 필요없거나 혹은 랜덤ID가 필요한 경우
`add()` 메서드를 호출하면 편리함

Firestore 단일문서

● ● ●

새로 추가 or 덮어쓰기

```
val city = hashMapOf(
    "name" to "Los Angeles",
    "state" to "CA",
    "country" to "USA"
)

db.collection("cities").document("LA")
    .set(city)
    .addOnSuccessListener { Log.d(TAG, "DocumentSnapshot successfully written!") }
    .addOnFailureListener { e -> Log.w(TAG, "Error writing document", e) }
```

업데이트 하는 내용

```
val dat = hashMapOf("capital" to true)

db.collection("cities").document("BJ")
    .set(dat, SetOptions.merge())
```

Firestore 단일문서 업데이트

전체 문서를 덮어쓰지 않고 일부 필드만 업데이트 하기 위해서는
update() 메서드를 사용해야함

데이터 하나만 업데이트

데이터 여러개 업데이트



```
db.collection("cities").document("DC")  
    .update("capital", true)  
  
db.collection("users").document("frank")  
    .update(mapOf(  
        "age" to 13,  
        "favorites.color" to "Red"  
    ))
```


Firestore 추가내용

배열요소 업데이트

- `FieldValue.arrayUnion()` -> 배열에 없는 요소 추가
- `FieldValue.arrayRemove()` -> 배열에 있는 요소 삭제

숫자값 늘리기

- `FieldValue.increment(n)` -> 숫자 필드 값 n만큼 증가

`update()` 메서드의 2번째 인자로 넣어주면 됨

** 자세한 사용법은 Firebase Firestore 공식 문서 참조

Firestore 문서



```
val washingtonRef = db.collection("cities").document("DC")  
  
washingtonRef.update("regions", FieldValue.arrayUnion("greater_virginia"))  
  
washingtonRef.update("regions", FieldValue.arrayRemove("east_coast"))  
  
washingtonRef.update("population", FieldValue.increment(50))
```

Firestore 문서 삭제

문서를 지우려면 delete() 메서드 사용



```
db.collection("cities").document("DC")
    .delete()
    .addOnSuccessListener { Log.d(TAG, "DocumentSnapshot successfully deleted!") }
    .addOnFailureListener { e -> Log.w(TAG, "Error deleting document", e) }
```

Firestore 필드 삭제

문서에서 특정 필드를 삭제하려면

업데이트 할 때 `FieldValue.delete()` 메서드 사용



```
val docRef = db.collection("cities").document("BJ")

val updates = hashMapOf<String, Any>(
    "capital" to FieldValue.delete()
)

docRef.update(updates).addOnCompleteListener { }
```

Firestore 컬렉션 삭제

안드로이드 앱에서 Firestore 컬렉션을 삭제하는 기능은 지원하지 않습니다.

Firestore 단일 문서 가져오기

document()를 통해 가져올 문서를 지정한 후
get() 메서드를 통해 단일 문서를 가져올 수 있습니다.

```
val docRef = db.collection("cities").document("SF")
docRef.get()
    .addOnSuccessListener { document ->
        if (document != null) {
            Log.d(TAG, "DocumentSnapshot data: ${document.data}")
        } else {
            Log.d(TAG, "No such document")
        }
    }
    .addOnFailureListener { exception ->
        Log.d(TAG, "get failed with ", exception)
    }
```

Firestore 여러 문서 가져오기

where() 를 통해 특정 조건을 만족하는 문서만 가져올 수 있음

where() 를 지우면 모든 문서를 가져올 수 있습니다.

```
db.collection("cities")
    .whereEqualTo("capital", true)
    .get()
    .addOnSuccessListener { documents ->
        for (document in documents) {
            Log.d(TAG, "${document.id} => ${document.data}")
        }
    }
    .addOnFailureListener { exception ->
        Log.w(TAG, "Error getting documents: ", exception)
    }
```


Firestore 정렬과 제한

orderBy() 를 통해 특정 필드를 기준으로 정렬할 수 있음

- 내림차순 정렬을 원한다면 orderBy의 2번째 인자로

Query.Direction.DESCENDING 을 넣어주면 됨.

limit() 을 통해 가져올 문서의 갯수를 제한할 수 있음



```
citiesRef.orderBy("name").limit(3)  
citiesRef.orderBy("name", Query.Direction.DESCENDING).limit(3)
```


더 자세한 내용은 공식 문서에서