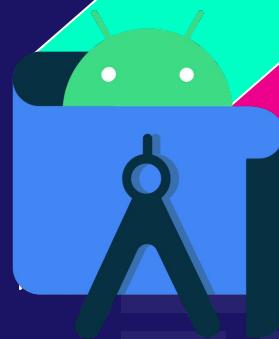


#13

Android Studio



# Android Studio

" Online Cloud Database,  
Common app solutions: Firebase "

# INNEHÅLLSFÖRTECKNING

01

Översikt

03

Firebase  
Persist & Read

02

Firebase Setup

04

Compose  
SideEffect

# INNEHÅLLSFÖRTECKNING



05



Övningar &  
Uppgifter



01

ÖVERSIKT

# Firestore the NOSQL cloud realtime DB



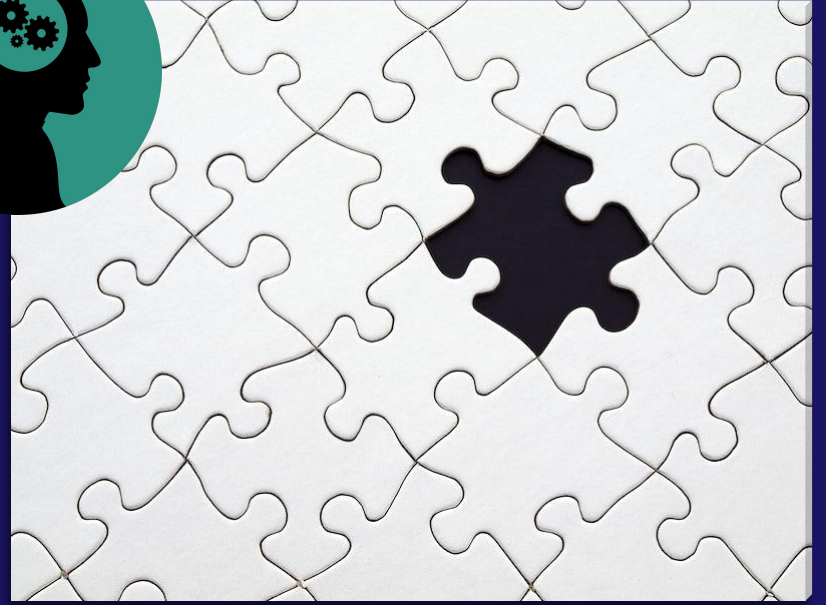
# Firebase

Används inte ROOM?

- Online / Cloud (Offline to disk works too)
- NoSQL
- Realtime Database
- Cannot export from Firebase  
(you're stuck with firebase if you use it)
- Easy and well documented process
- Quick setup



*Frågor?*





02

# Firebase Setup



# Firestore RESTRICTIONS



ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.0 Ice Cream Sandwich	15	
4.1 Jelly Bean	16	99.8%
4.2 Jelly Bean	17	99.2%
4.3 Jelly Bean	18	98.4%
4.4 KitKat	19	98.1%
5.0 Lollipop	21	94.1%
5.1 Lollipop	22	92.3%
6.0 Marshmallow	23	84.9%
7.0 Nougat	24	73.7%
7.1 Nougat	25	66.2%
8.0 Oreo	26	60.8%
8.1 Oreo	27	53.5%
9.0 Pie	28	39.5%
10, Android 10	29	8.2%

# FireBase API LEVEL

KitKat API level 19

- Targets API level 19 (KitKat) or higher

# FireBase Criteria

## Övriga kriterier:

- Uses Android 4.4 or higher
- Använder sig av Jetpack, som inkluderar följande kriterier:
  - `com.android.tools.build:gradle v3.2.1` eller senare
  - `compileSdkVersion 28` eller senare

*Glöm inte att logga in med ditt Google konto*

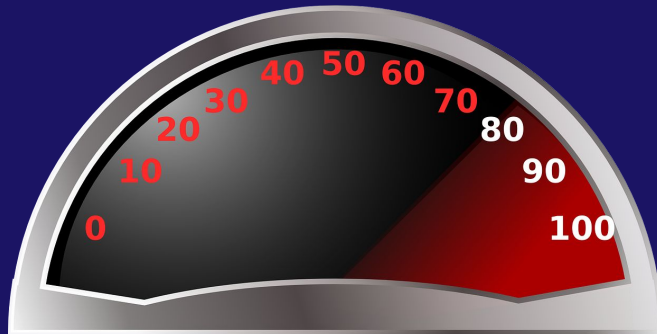


# Firestore Limitations



# FireBase Limitations

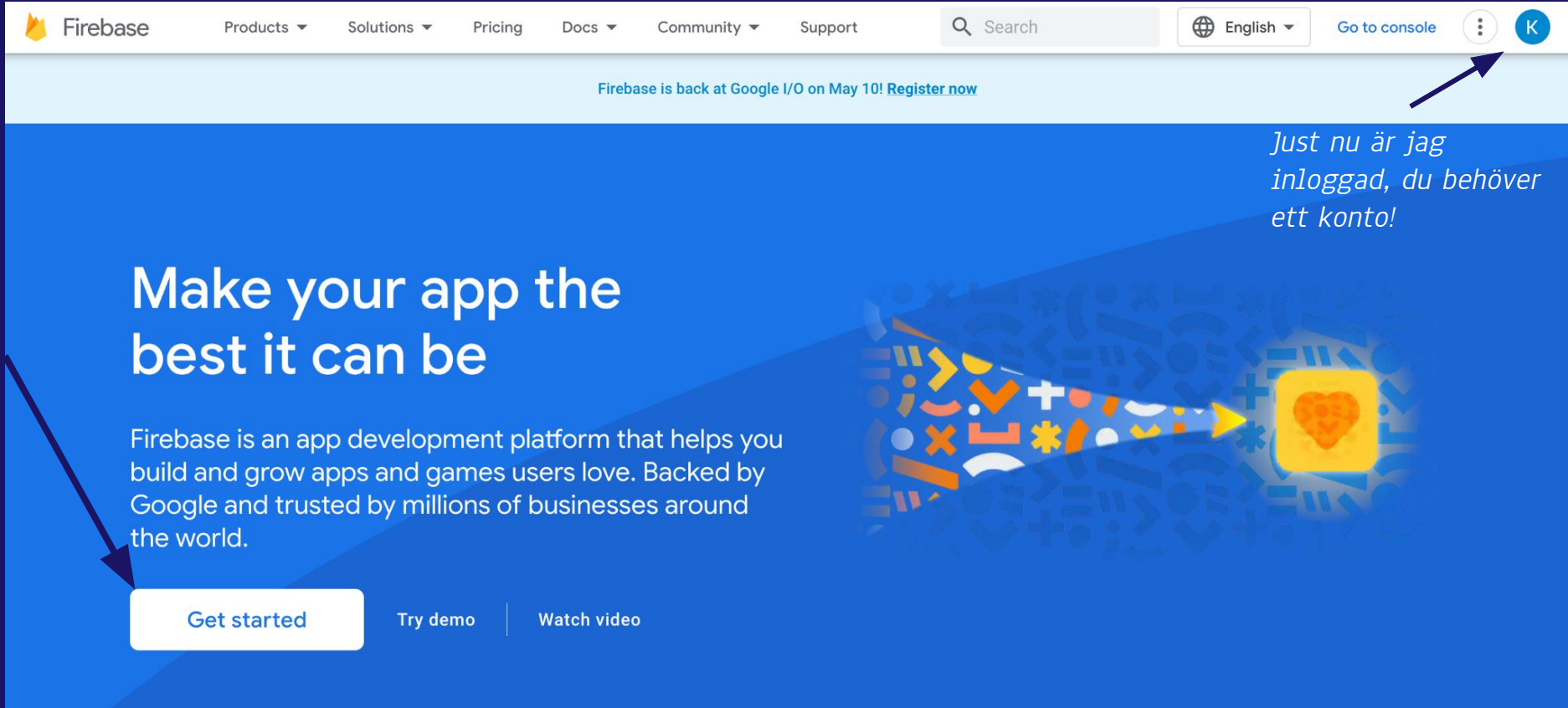
<https://firebase.google.com/docs/database/usage/limits>



# Firebase: How to



# Sign up & get started



The image is a screenshot of the Firebase website's homepage. The header is white with the Firebase logo on the left and navigation links (Products, Solutions, Pricing, Docs, Community, Support) in the center. On the right, there is a search bar, a language selector (English), a 'Go to console' link, and a user profile icon with the letter 'K'. A blue banner below the header contains the text 'Firebase is back at Google I/O on May 10! Register now'. The main content area has a blue background with the headline 'Make your app the best it can be'. Below this, a paragraph describes Firebase as an app development platform. At the bottom, there are three buttons: 'Get started', 'Try demo', and 'Watch video'. An illustration on the right shows a yellow heart icon being targeted by a beam of light from a pattern of colorful symbols. Two blue arrows are overlaid on the image: one points from the 'Get started' button to the main text area, and the other points from the user profile icon in the header to the text 'Just nu är jag inloggad, du behöver ett konto!'.

Products ▾ Solutions ▾ Pricing Docs ▾ Community ▾ Support

Search

English ▾ Go to console

K

Firebase is back at Google I/O on May 10! [Register now](#)

## Make your app the best it can be

Firebase is an app development platform that helps you build and grow apps and games users love. Backed by Google and trusted by millions of businesses around the world.

[Get started](#) [Try demo](#) [Watch video](#)

Just nu är jag inloggad, du behöver ett konto!

# Add Project +

Your Firebase projects



Add project

Lägg till!



# Naming

× Create a project (Step 1 of 3)

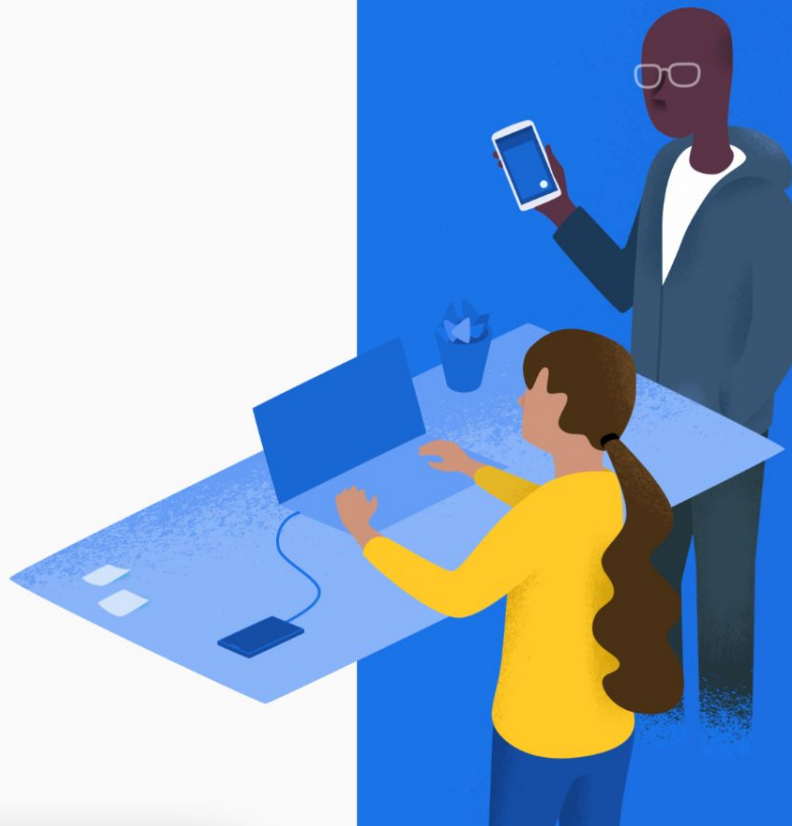
Let's start with a name for  
your project<sup>?</sup>

Project name

Demo-13

 fir-12-b2350

Continue



# Google Analytics

× Create a project (Step 2 of 2)

## Google Analytics for your Firebase project

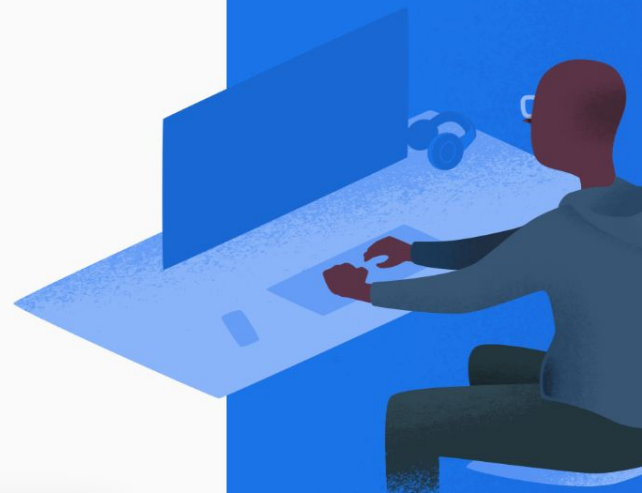
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Google Analytics enables:


- × A/B testing ?
- × User segmentation & targeting across Firebase products ?
- × Crash-free users ?
- × Event-based Cloud Functions triggers ?
- × Free unlimited reporting ?




Enable Google Analytics for this project  
Recommended





# Android


 **Firebase**


[Project Overview](#) 


Product categories

[Build](#) 

[Release & Monitor](#) 

[Analytics](#) 

[Engage](#) 







 All products

**Customize your nav!**  
You can now focus your console experience by customizing your navigation  
[Learn more](#) [Got it](#)


demo-13 ▾

demo-13 [Spark plan](#)

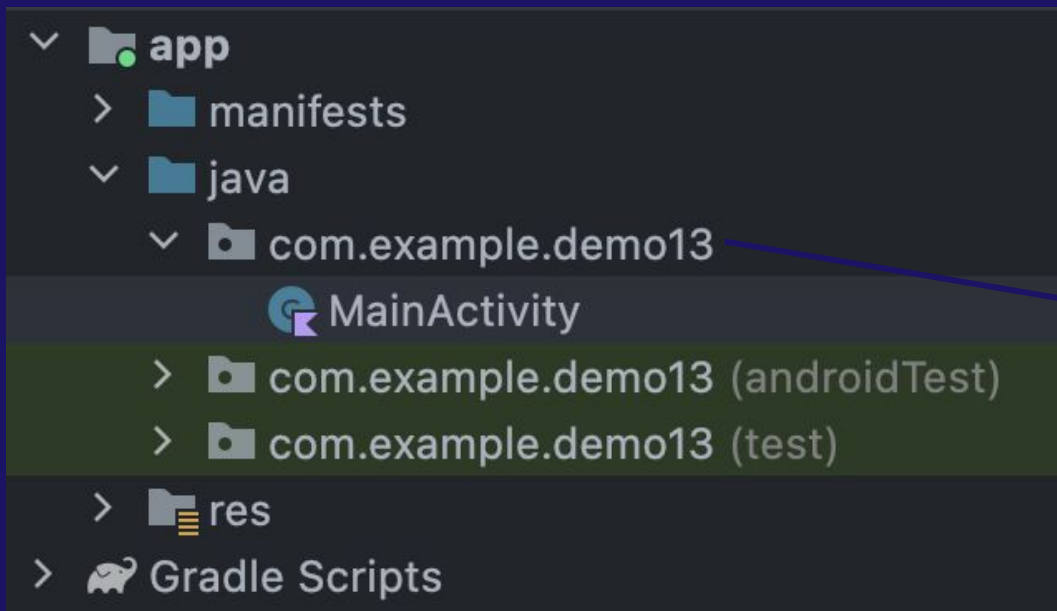
Get started by adding  
Firebase to your app



Add an app to get started



# Package



## Add Firebase to your Android app

### 1 Register app

Android package name ?

Detta MÅSTE matcha!

[illegible]

← Android Studio Firebase Demo

← BLANK

## Register app

Android package name 

com.example.demo13

App nickname (optional) 

## My Android App

Debug signing certificate SHA-1 (optional) ?

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:(

---

**i** Required for Dynamic Links, and Google Sign-In or phone number support in Auth.  
Edit SHA-1s in Settings.

## Register app



## Register app

Android package name 

com.example.demo13

App nickname (optional) 

# My Android App

Debug signing certificate SHA-1 (optional) 

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:(

**i** Required for Dynamic Links, and Google Sign-In or phone number support in Auth.  
Edit SHA-1s in Settings.

## Register app





## Register app

Android package name: com.example.demo13, App nickname: My Android App

2

## Download and then add config file

Instructions for Android Studio below | [Unity](#) [C++](#)



Download google-services.json

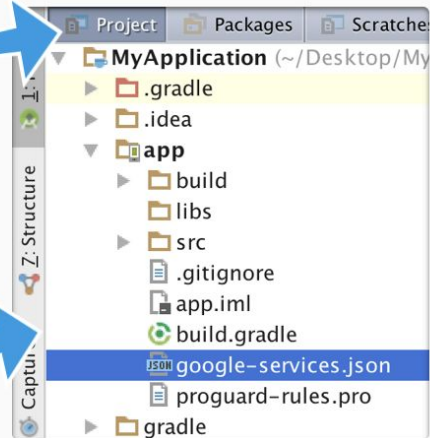
Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.



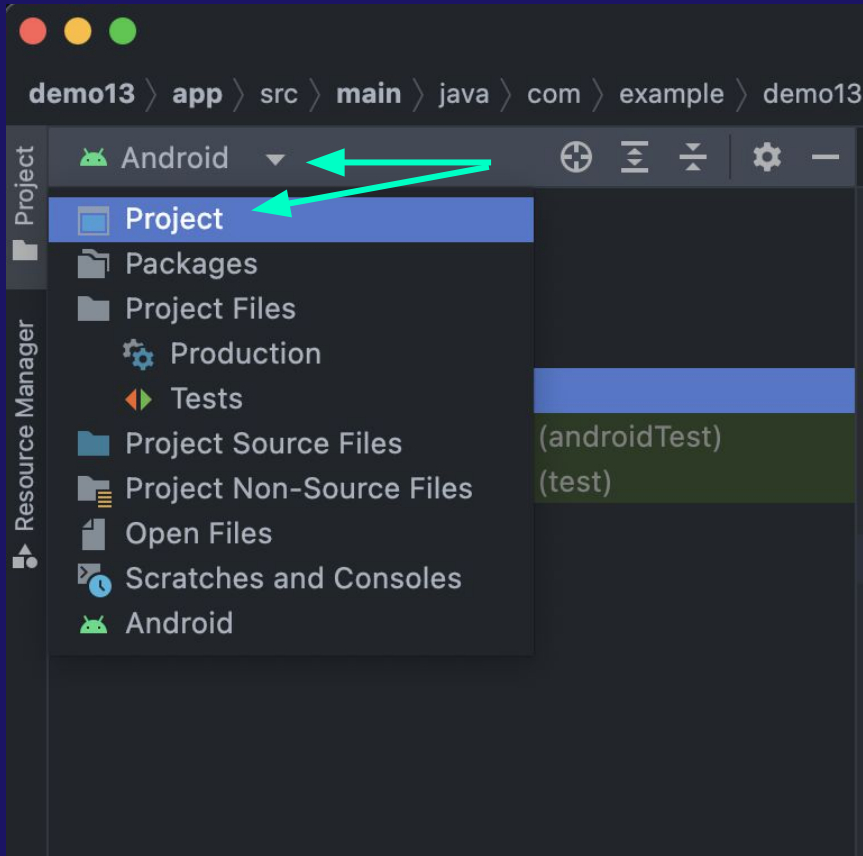
google-services.json

Next

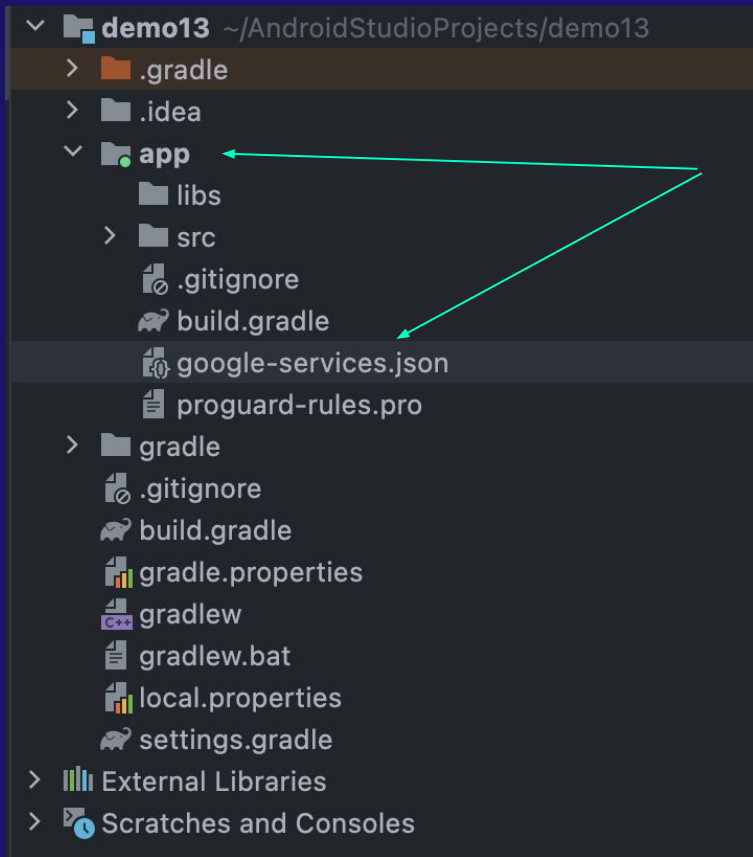




# Project View





# Drag And Drop





# Project Gradle


## ✓ Gradle Scripts

 build.gradle (Project: demo13)


 build.gradle (Module :app)

 proguard-rules.pro (ProGuard Rules for ":app")

 gradle.properties (Project Properties)

 gradle-wrapper.properties (Gradle Version)

 local.properties (SDK Location)

 settings.gradle (Project Settings)

# Dependencies


```
// Top-level build file where you can add configuration options
common to all sub-projects/modules.
buildscript {
    repositories {
        // Make sure that you have the following two repositories
        google() // Google's Maven repository
        mavenCentral() // Maven Central repository
    }


    dependencies {
        // Add the dependency for the Google services Gradle plugin
        classpath 'com.google.gms:google-services:4.3.15'
    }
}


plugins {
    id 'com.android.application' version '7.4.2' apply false
    id 'com.android.library' version '7.4.2' apply false
    id 'org.jetbrains.kotlin.android' version '1.8.0' apply false
}
```

# Setup


## ▼ Gradle Scripts

 build.gradle (Project: demo13)


 build.gradle (Module :app)

 proguard-rules.pro (ProGuard Rules for ":app")

 gradle.properties (Project Properties)

 gradle-wrapper.properties (Gradle Version)

 local.properties (SDK Location)

 settings.gradle (Project Settings)

# Plugin

```
plugins {  
    id 'com.android.application'  
    id 'org.jetbrains.kotlin.android'  
    id 'com.google.gms.google-services' ←  
}
```

# Dependency

Dependency!

```
// Import the Firebase BoM
implementation platform('com.google.firebase:firebase-bom:31.2.3' )

// TODO: Add the dependencies for Firebase products you want to use
// When using the BoM, don't specify versions in Firebase
dependencies
// https://firebase.google.com/docs/android/setup#available-libraries
```

# Setup Complete

## 4 Next steps

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

[Previous](#)

[Continue to console](#)

[Documentation](#) <https://firebase.google.com/docs/guides?authuser=0&hl=en>

[Sample apps](#): <https://firebase.google.com/docs/samples?authuser=0&hl=en>



# Restriction!

Gradle builds that use Android Gradle plugin (AGP) v4.2 or earlier need to enable Java 8 support. Otherwise, these Android projects get a build failure when adding a Firebase SDK.

To fix this build failure, you can follow one of two options:

- Add the listed `compileOptions` from the error message to your **app-level** `build.gradle` file.
- Increase the `minSdkVersion` for your Android project to 26 or above.

# What's next?

Add Firebase services to your app:

- Gain insights on user behavior with [Analytics](#).
- Set up a user authentication flow with [Authentication](#).
- Store data, like user information, with [Cloud Firestore](#) or [Realtime Database](#).
- Store files, like photos and videos, with [Cloud Storage](#).
- Trigger backend code that runs in a secure environment with [Cloud Functions](#).
- Send notifications with [Cloud Messaging](#).
- Find out when and why your app is crashing with [Crashlytics](#).

# Realtime Database Setup



# Realtime DB Dependency

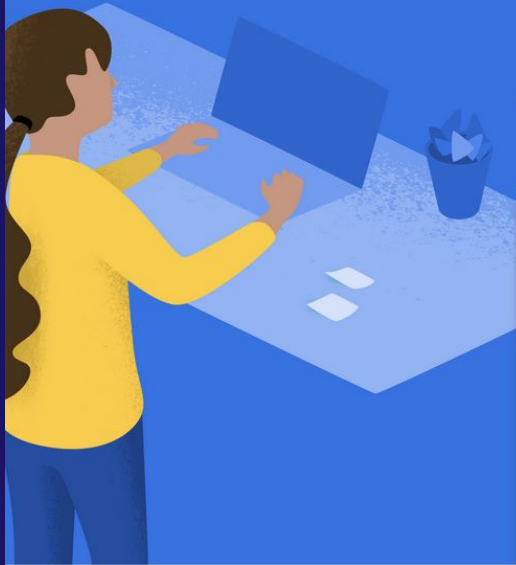
Vi kommer sätta upp en 'Realtime Database'

```
// Add the dependency for the Realtime Database library
// When using the BoM, you don't specify versions in Firebase library dependencies
implementation 'com.google.firebase:firebase-database-ktx'
```

# Enter Project



Firebase



Your Firebase projects



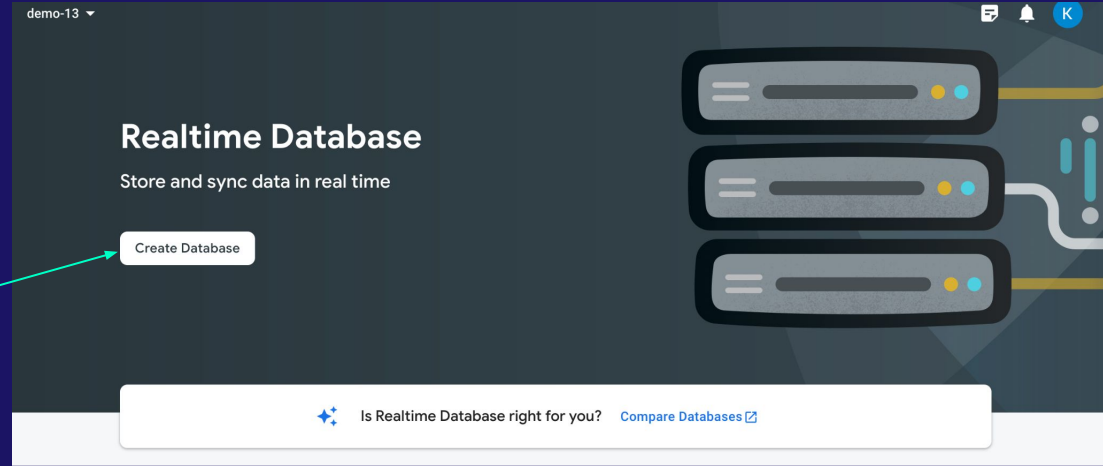
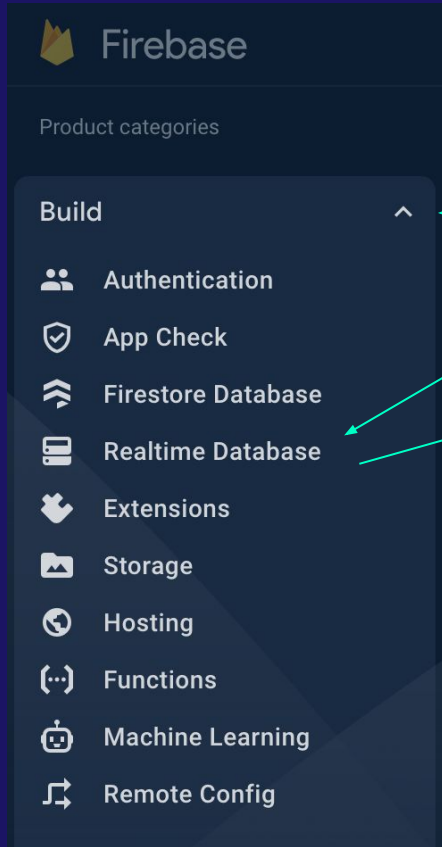
Add project

<https://console.firebase.google.com/>

**demo-13**

fir-13-d9c11

# Realtime DB



# Europe

## Set up database



1

Database options

2

Security rules

Your location setting is where your Realtime Database data will be stored.

Realtime Database location

Belgium (europe-west1)



Cancel

Next

# Test Mode

## Set up database



Database options



Security rules

Once you have defined your data structure **you will have to write rules to secure your data.**

[Learn more](#)



### Start in **locked mode**

Your data is private by default. Client read/write access will only be granted as specified by your security rules.



### Start in **test mode**

Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

```
{
  "rules": {
    ".read": "now < 1682028000000", // 2023-4-21
    ".write": "now < 1682028000000", // 2023-4-21
  }
}
```



**The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days**

Cancel

Enable



# Test Mode

## Test mode

Good for getting started with the mobile and web client libraries, but allows anyone to read and overwrite your data. After testing, **make sure to review the [Understand Firebase Realtime Database Rules](#) section.**

★ **Note:** If you create a database in Test mode and make no changes to the default world-readable and world-writable Rules within a trial period, you will be alerted by email, then your database rules will deny all requests. Note the expiration date during the Firebase console setup flow.

To get started with the web, Apple, or Android SDK, select testmode.

## Locked mode

Denies all reads and writes from mobile and web clients. Your authenticated application servers can still access your database.



03

Read & Write  
to Firebase

# Setup



# Initialize

```
package com.example.demo13





import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.google.firebase.database. DatabaseReference

class MainActivity : AppCompatActivity () {

    private lateinit var db : DatabaseReference

    override fun onCreate(savedInstanceState : Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

# Link

**Firebase**  
 Firestore Database  
 Realtime Database  
  
Product categories  
  
**Build** ▾  
**Release & Monitor** ▾  
**Analytics** ▾  
**Engage** ▾  
  
 All products  
  
**Customize your nav!**  
You can now focus your console experience by customizing your navigation  
[Learn more](#) [Got it](#)  
  
**Spark**  
No-cost \$0/month **Upgrade**

demo-13 ▾


**Realtime Database**

?

[Data](#) [Rules](#) [Backups](#) [Usage](#) [Extensions](#) **NEW**

Kopiera länk!

Protect your Realtime Database resources from abuse, such as billing fraud or phishing [Configure App Check](#) ×

 <https://europe-west1.firebaseio.com> ⌵ ⌵ ⋮

# URL

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    setContentView(R.layout.activity_main)  
  
    db = FirebaseDatabase  
        .getInstance("YOUR_URL_HERE")  
        .getReference("users")  
  
}
```

# URL



User

```
package com.example.demo13

data class User(
    val name: String? = null,
    val password: String? = null
) {

}
```

# Database Operations

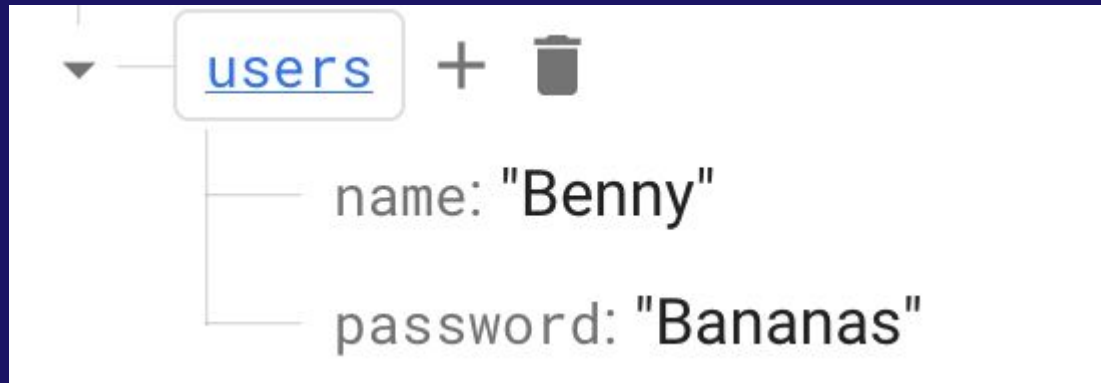





# Push USER

```
db.setValue(  
    User("Benny", "Bananas")  
)  
.addOnSuccessListener {  
    Toast.makeText(this, "Pushed user", Toast.LENGTH_LONG).show()  
  
}.addOnFailureListener {  
    Toast.makeText(this, "FAILURE", Toast.LENGTH_LONG).show()  
  
}
```

# Persisted!

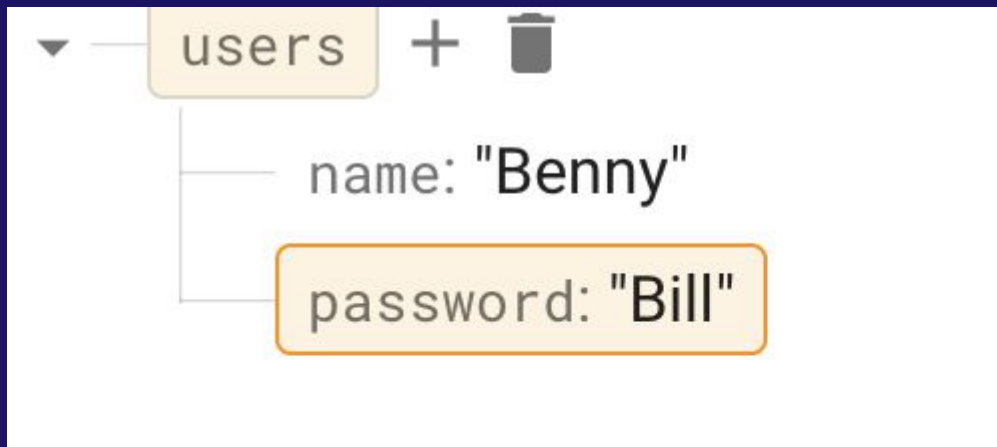


# Update User



```
db.setValue(  
    User( name: "Benny", password: "Bill")  
) .addOnSuccessListener { it: Void!  
    Toast.makeText( context: this, text: "Pushed user", Toast.LENGTH_LONG).show()  
} .addOnFailureListener { it: Exception  
    Toast.makeText( context: this, text: "FAILURE", Toast.LENGTH_LONG).show()  
}
```

# Updated!



# Design

```
// ID's
```

```
val teUsername = findViewById<EditText>(R.id.et_username)
```

```
val tePassword = findViewById<EditText>(R.id.et_password)
```

```
val tvDb = findViewById<TextView>(R.id.tv_db)
```

```
val btnSubmitUser = findViewById<Button>(R.id.btn_submit)
```

username

password

Hello World!

BUTTON

# Listen for Changes

Lyssnare för databasändringar

```
val userListener = object : ValueEventListener {  
    override fun onDataChange(snapshot: DataSnapshot) {  
        TODO("Not yet implemented")  
    }  
  
    override fun onCancelled(error: DatabaseError) {  
        TODO("Not yet implemented")  
    }  
}
```

# Listen For Changes

```
private fun addUserEventListener (
    userReference: DatabaseReference,
    textView: TextView) {

    val userListener = object : ValueEventListener {

        override fun onDataChange (snapshot: DataSnapshot) {
            val user = snapshot.getValue<User>()

            textView.text = user.toString()
        }

        override fun onCancelled (error: DatabaseError) {
            println(error)
        }
    }

    userReference.addValueEventListener (userListener)
}
```

Vi passar in databasen + textvyn som vi vill ska visa upp vår NYA förändring

# Design

```
private fun addUser(db: DatabaseReference, context: Context,
username: String, password: String) {
    db.setValue(
        User(username, password)
    ).addOnSuccessListener {
        Toast.makeText(context, "Pushed user $username $password",
Toast.LENGTH_LONG).show()

    }.addOnFailureListener {
        Toast.makeText(context, "FAILURE", Toast.LENGTH_LONG).show()
    }
}
```

username

password

Hello World!

BUTTON



# OnCreate()

FETCH MM.. TLILFÖR SENARE - TACK

```
// ID's
val tvDb = findViewById<TextView>(R.id.tv_db)
val teUsername = findViewById<EditText>(R.id.et_username)
val tePassword = findViewById<EditText>(R.id.et_password)
val btnSubmitUser = findViewById<Button>(R.id.btn_submit)

db = FirebaseDatabase

.getInstance("https://fir-13-d9c11-default-rtdb.europe-west1.firebaseio.com/")
    .getReference("users")

btnSubmitUser.setOnClickListener {
    addUser(db, this, teUsername.text.toString(), tePassword.text.toString())
}

addUserEventListener(db, tvDb)
```

# Fetch User

```
// On Fetch
btnFetchUser.setOnClickListener { it: View!
    db.child( pathString: "-NRXYny37FJF8btRkYPx") DatabaseReference
        .get() Task<DataSnapshot!>
        .addOnSuccessListener { it: DataSnapshot!
            // val newUser: User = it

            val newUser = User(
                it.child( path: "username").value.toString(),
                it.child( path: "password").value.toString(),
                it.child( path: "isRegistered").value.toString().toBoolean()
            )

            tvUser.text = newUser.username
        }

        .addOnFailureListener { it: Exception
            Toast.makeText(applicationContext, text: "An error occurred: $it", Toast.LENGTH_LONG).show()
        }
    }
```

# Fetch User

```
/* Info
 * db.orderByChild("users")          <-- Fetches ALL users
 * db.child("-NRXYny37FJF8btRkYPx") <-- Fetches ID
 * */
```

```
// On Fetch
btnFetchUser.setOnClickListener { it: View!
    db.child( pathString: "-NRXYny37FJF8btRkYPx") DatabaseReference
        .get() Task<DataSnapshot!>
        .addOnSuccessListener { it: DataSnapshot!
            // val newUser: User = it

            val newUser = User(
                it.child( path: "username").value.toString(),
                it.child( path: "password").value.toString(),
                it.child( path: "isRegistered").value.toString().toBoolean()
            )

            tvUser.text = newUser.username
        }

        .addOnFailureListener { it: Exception
            Toast.makeText(applicationContext, text: "An error occurred: $it", Toast.LENGTH_LONG).show()
        }
}
```



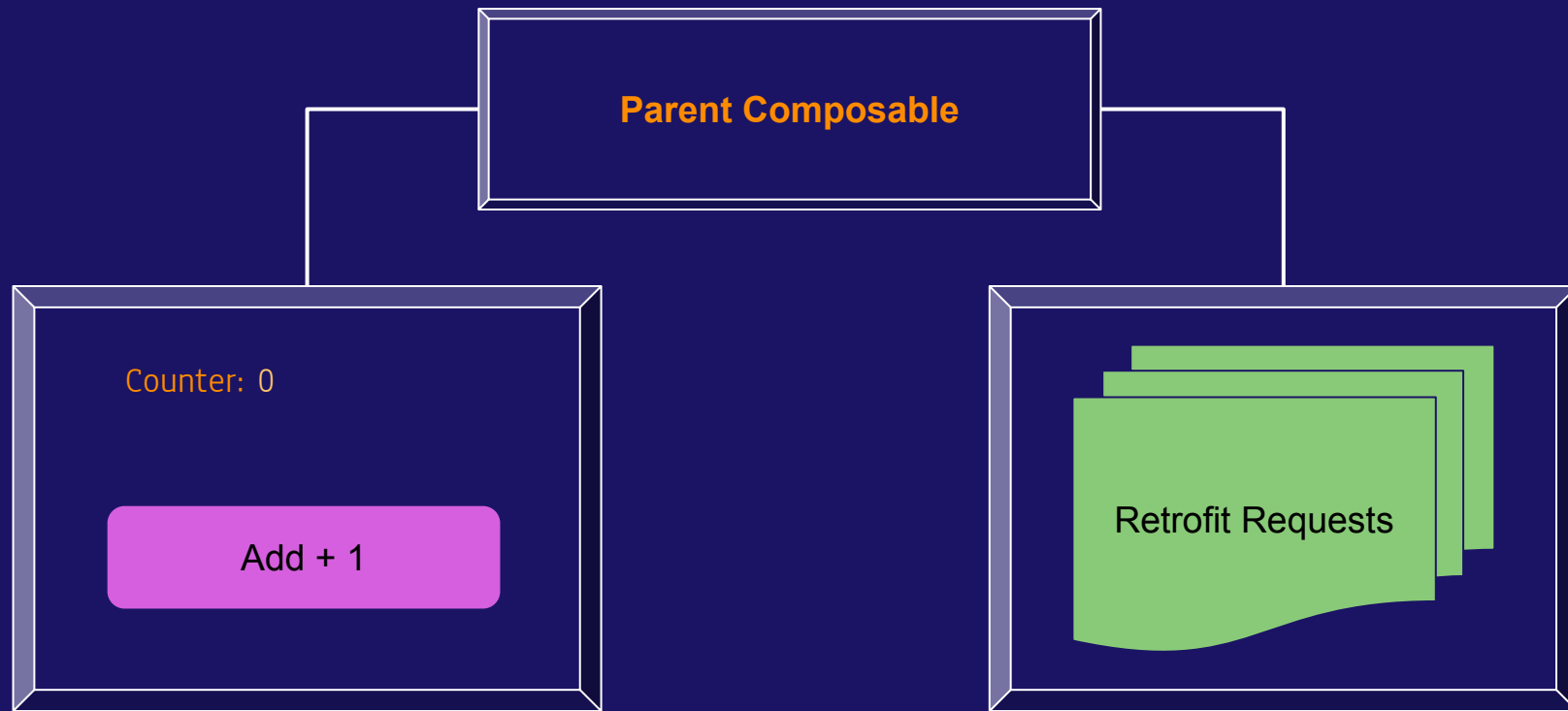
04

Compose SideEffect

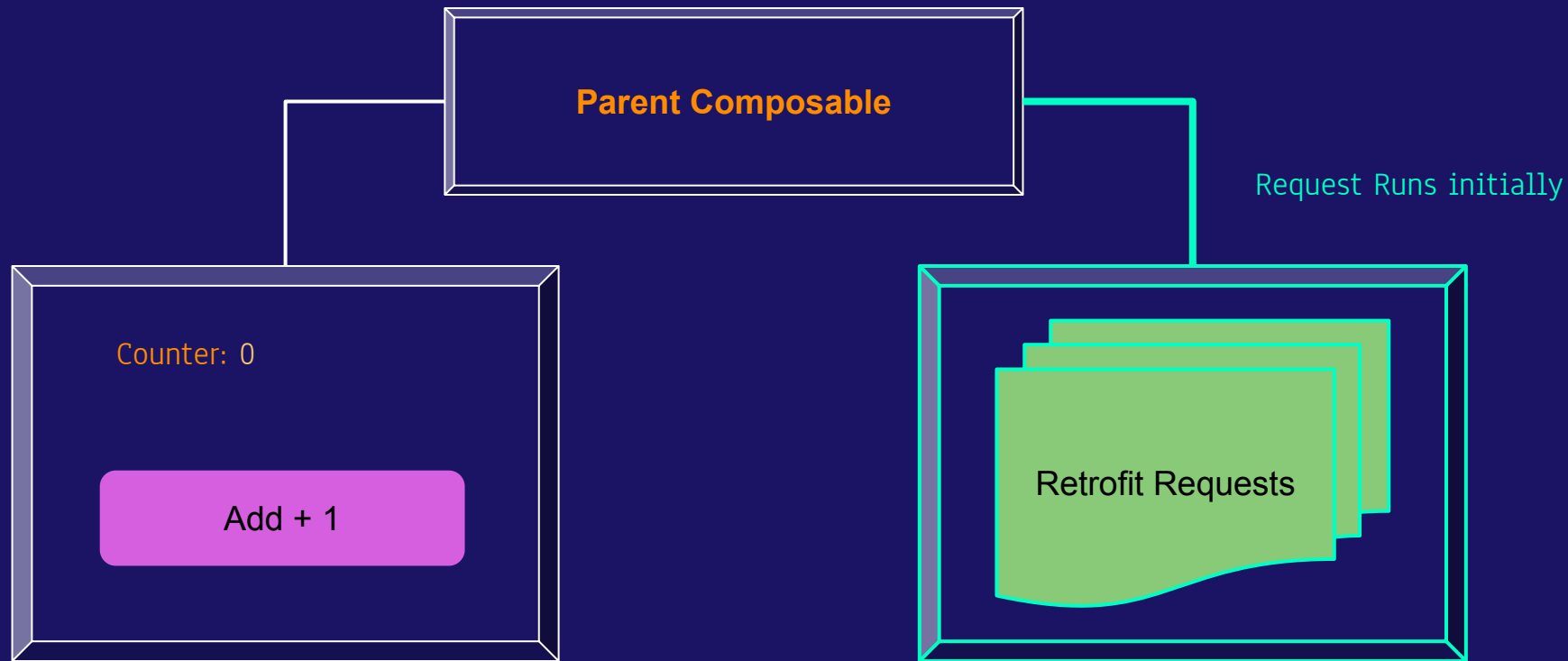
# SideEffects?



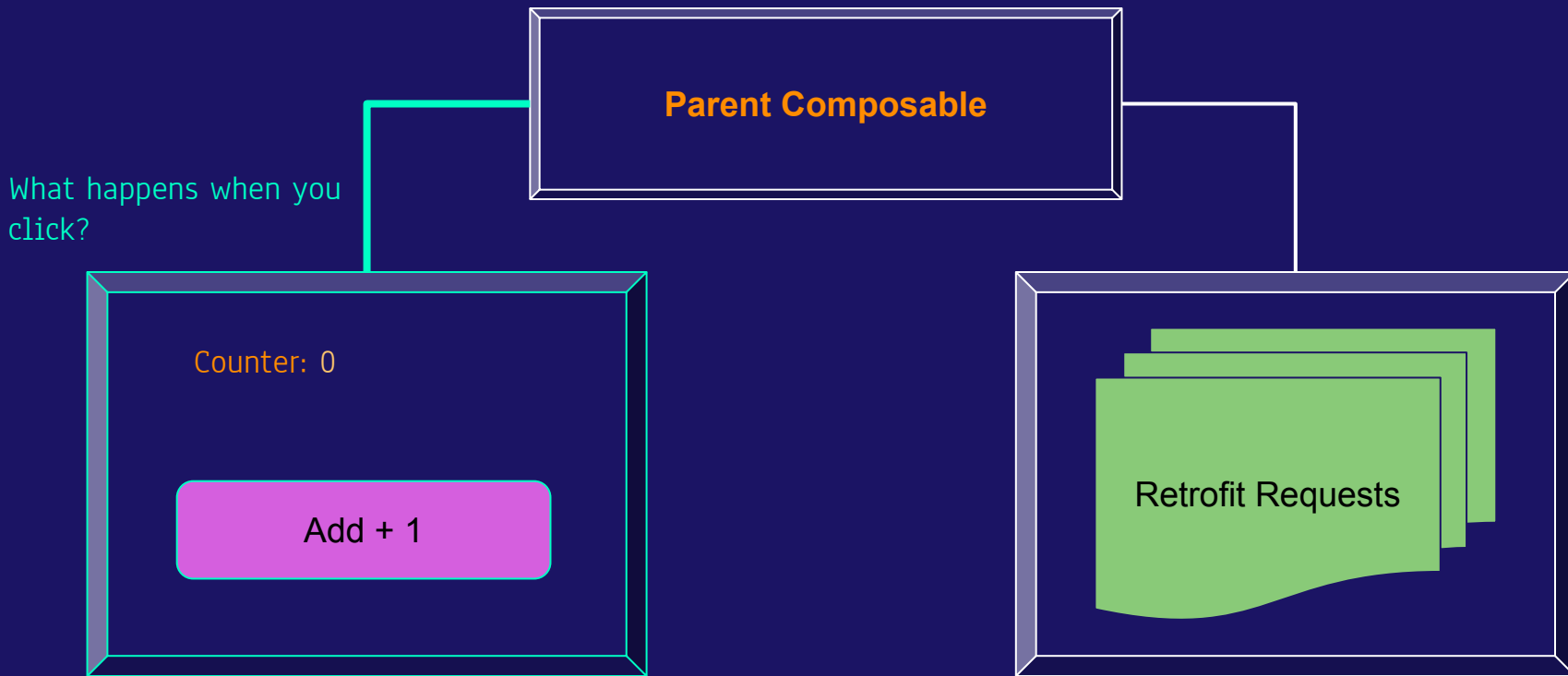
# 3 Composables



# First Step

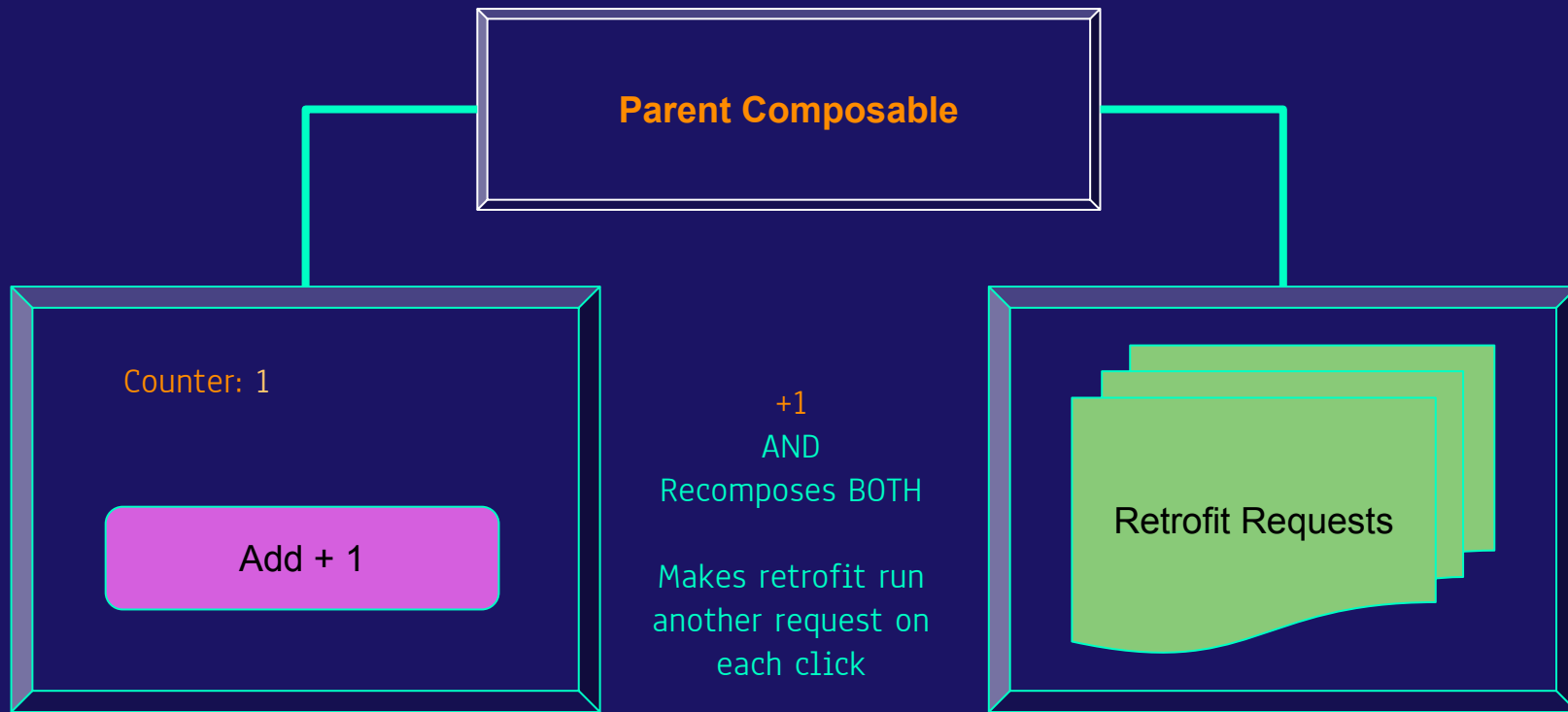


# 3 Composables





# 3 Composables



# Example LOOP

```
@Composable
fun ParentScreen () {
    val parentCounter = remember { mutableStateOf (0) }
    val childCounter = remember { mutableStateOf (0) }

    SideEffect {
        // Update the parent and child counters
        parentCounter.value++
        childCounter.value++
    }

    Column {
        Text(text = "Parent counter: ${parentCounter.value}")
        ChildScreen(childCounter.value)
    }
}
```

## SideEffect: publish Compose state to non-compose code

To share Compose state with objects not managed by compose, use the `SideEffect` composable, as it's invoked on every successful recomposition.

For example, your analytics library might allow you to segment your user population by attaching custom metadata ("user properties" in this example) to all subsequent analytics events. To communicate the user type of the current user to your analytics library, use `SideEffect` to update its value.

```
@Composable
fun rememberFirebaseAnalytics(user: User): FirebaseAnalytics {
    val analytics: FirebaseAnalytics = remember {
        FirebaseAnalytics()
    }

    // On every successful composition, update FirebaseAnalytics with
    // the userType from the current User, ensuring that future analytics
    // events have this metadata attached
    SideEffect {
        analytics.setUserProperty("userType", user.userType)
    }
    return analytics
}
```

SideEffectsSnippets.kt

Source:

<https://developer.android.com/jetpack/compose/side-effects>

# Analytics Explained

The `rememberFirebaseAnalytics` function uses the `remember` function to create an instance of `FirebaseAnalytics` and return it. The `remember` function ensures that the same instance is returned during every recomposition of the composable function.

The `SideEffect` function is called during every successful recomposition of the composable function, but it doesn't run the `setUserProperty` function during every recomposition. It only runs when there is a change in the `user` parameter, because `user` is passed as a dependency to `SideEffect`. This means that the `setUserProperty` function is called only when the `user` parameter changes.

By using `SideEffect` in this way, we can ensure that the metadata attached to future analytics events is updated only when necessary, without causing unnecessary loops. In other words, the `SideEffect` function runs only when its dependencies change.

This example demonstrates how `SideEffect` can be used to run code during recomposition only when necessary, without causing unnecessary loops.

# 3 Composables

Tänk om vi har ett API som använder sig av en nyckel.

Vi får högst bara skapa 1000 requests om månaden.

Men varje knapptryck skapar oundvikligt ett nytt request.

- Onödigt mycket bandbredd används
- Sämre prestanda på applikationen
- Använder onödiga requests och tryck på servrar



# Run ONCE

```
@Composable
fun MyComposable () {
    val someValue = remember { mutableStateOf(0) }

    LaunchedEffect(true) { // pass in a unique value to run the effect once
        val response = makeNetworkRequest ()
        // update someValue or do other side effects based on the network response
    }

    // rest of the composable
}
```



05

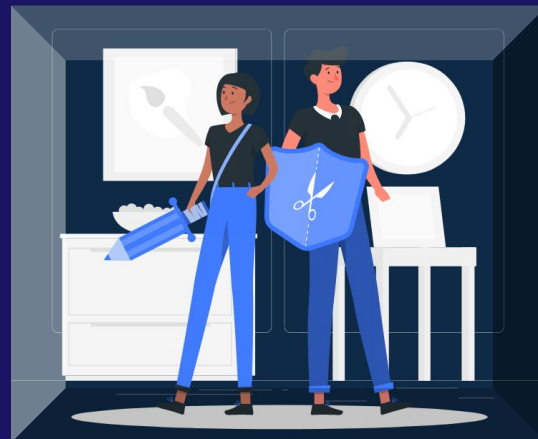
Uppgifter  
&  
Eget Arbete

## Välkommen till första uppgiften!

Uppgifterna är till för att testa dina färdigheter och kunskaper för att både öva och repetera på det vi har arbetat med under föreläsningarna.

Dessa är **INTE** obligatoriska.  
Men är starkt rekommenderat att arbeta med.

## Uppgifter





# MINNS DU?

```
// Vad finns för skillnader mellan  
Firebase och Room?
```

```
// Vad innebär NoSQL?
```

```
// Vilka fördelar finns med Firebase?
```

```
// Vilka nackdelar finns med Firebase?
```

```
// Varför lägger vi till en  
'valueEventListener' ?
```

```

1          // -Uppgift #1- //
2
3      /* INSTRUCTIONS
4
5          Skapa ett nytt projekt!
6
7          Döp projektet till: Lektion_13_uppgifter
8
9          Skapa nu en ny klass: 'Student'
10         +   name: String
11         +   age: Int
12
13         Skapa en enkel design:
14         +   Button Submit Student
15         +   Text Edit name
16         +   Text Edit schoolName
17         +   Text View DB Student
18     */
19
20     // HINT & Examples
21     hint(" Slide #47 hjälper dig med design ")
22
23

```

# Uppgift #1

*Kom igång enkelt med uppgift #1*

```

1      // -Uppgift #2- //
2
3      /* INSTRUCTIONS
4
5         Initialisera ett nytt projekt inom Firebase.
6
7         Följ stegen inom Slide #11
8
9         Skapa sedan en instans av din db som du
10        sedan kan koppla:
11
12        private lateinit var db : DatabaseReference
13        db = FirebaseDatabase
14            .getInstance("URL")
15            .getReference("users")
16
17        */
18
19
20
21
22
23

```

## Uppgift #2

*Följ instruktionerna inom Firebase.  
Glöm inte att 'package' ska vara  
samma både inom Firebase och  
inom ditt projekt!*

```
1          // -Uppgift #3- //
2
3      /* INSTRUCTIONS
4
5          Försök nu lägga till och läsa data från
6          Firebase via applikationen!
7
8      */
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```

## Uppgift #3

Slide #43 - Push user

Slide #45 - Update User

Slide #48 - Changes Listener

# THANKS!

Do you have any questions?  
[kristoffer.johansson@sti.se](mailto:kristoffer.johansson@sti.se)

[sti.learning.nu/](https://sti.learning.nu/)

CREDITS: This presentation template was created by  
Slidesgo, including icons by Flaticon, and  
infographics & images by Freepik.

*You can also contact me VIA Teams (quicker response)*  
*Du kan också kontakta mig VIA Teams (Snabbare svar)*