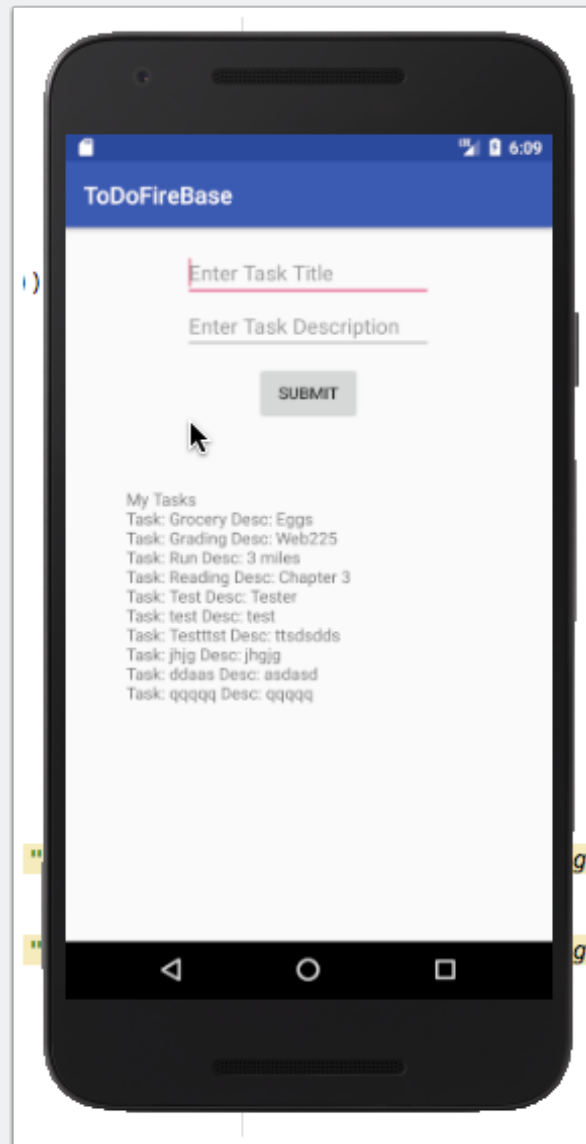


Android Emulator - ToDoList using Firebase

Github Link: <https://github.com/RVCAndroidClass/TaskFirebase>

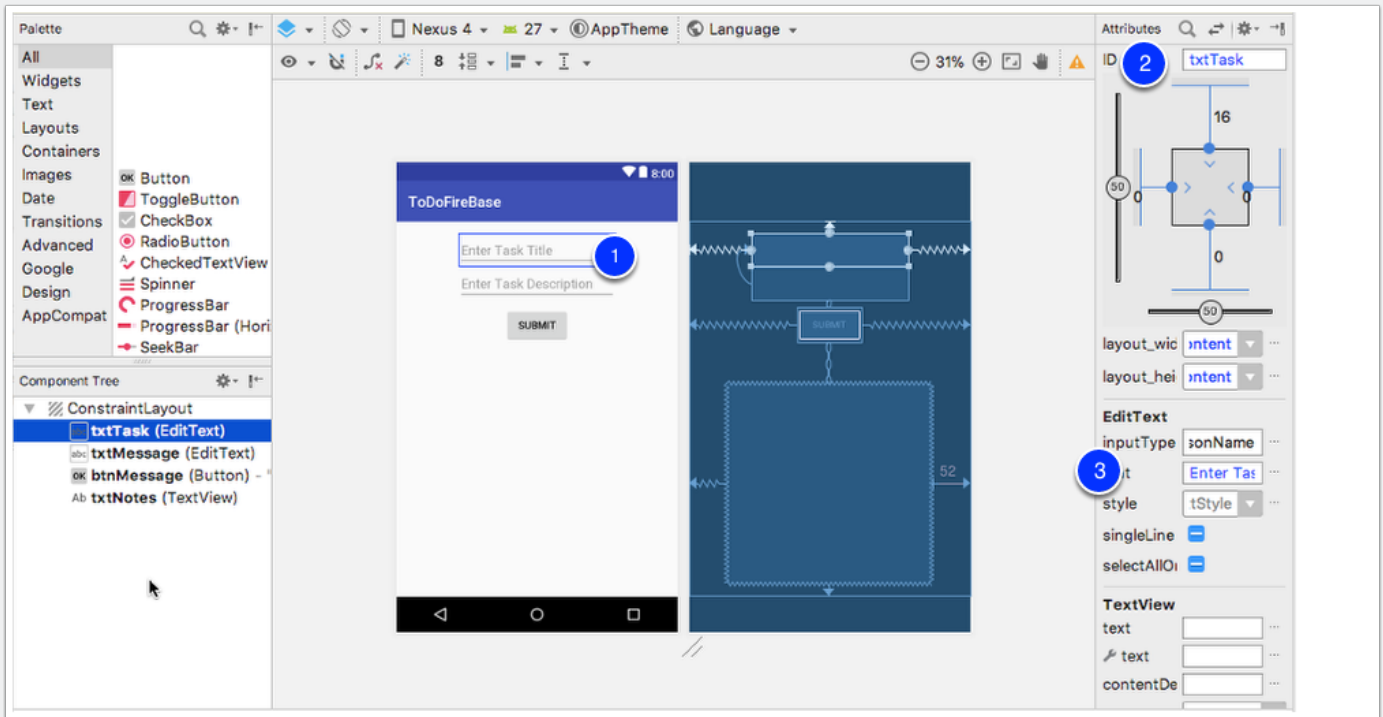
1. Go to <http://gmail.com>
2. Create a NEW Gmail Account Just For App
3. Logoff of Google in all of your web browsers (go to <http://google.com> in all your browsers)



Create New Project Named MyTodoList

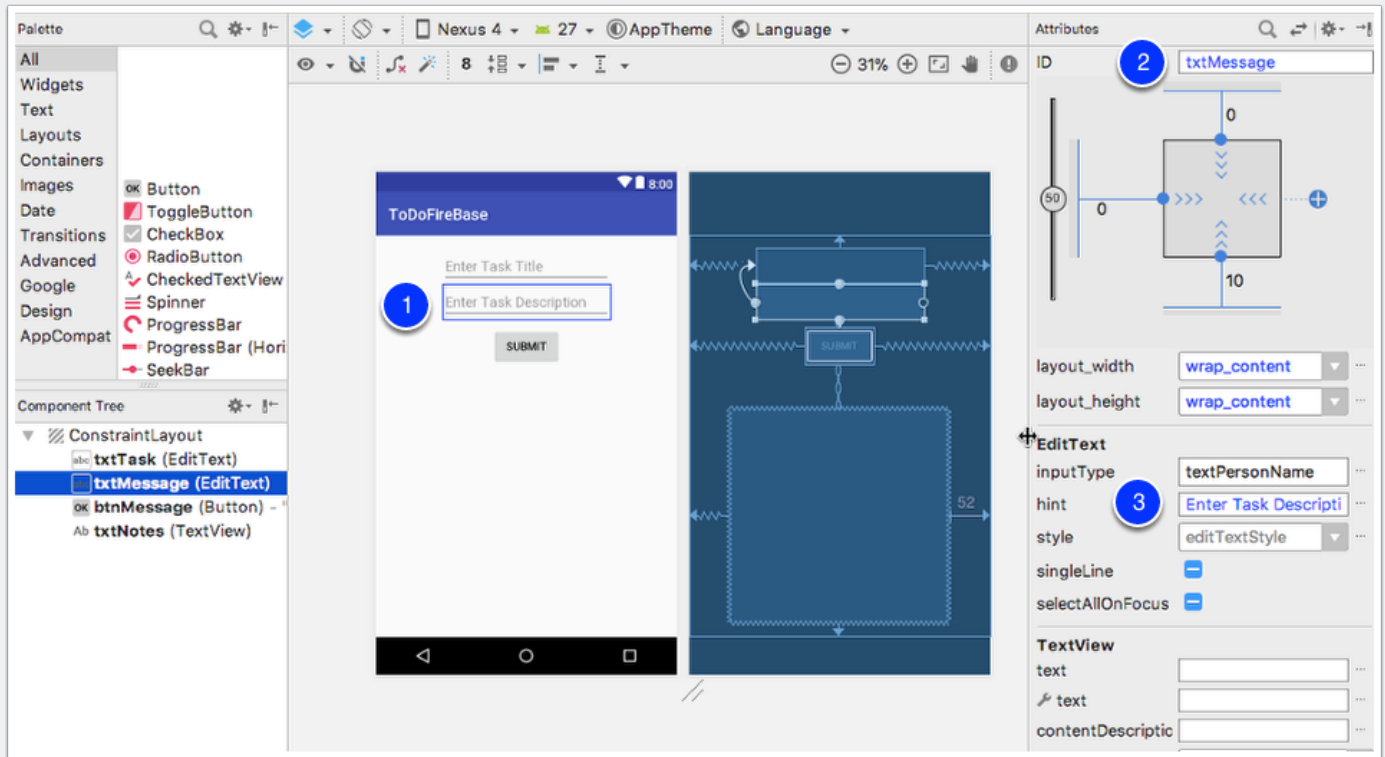
1. Remove Hello Label
2. Add EditText,
3. ID: txtTask

4. Hint: Enter Task Name



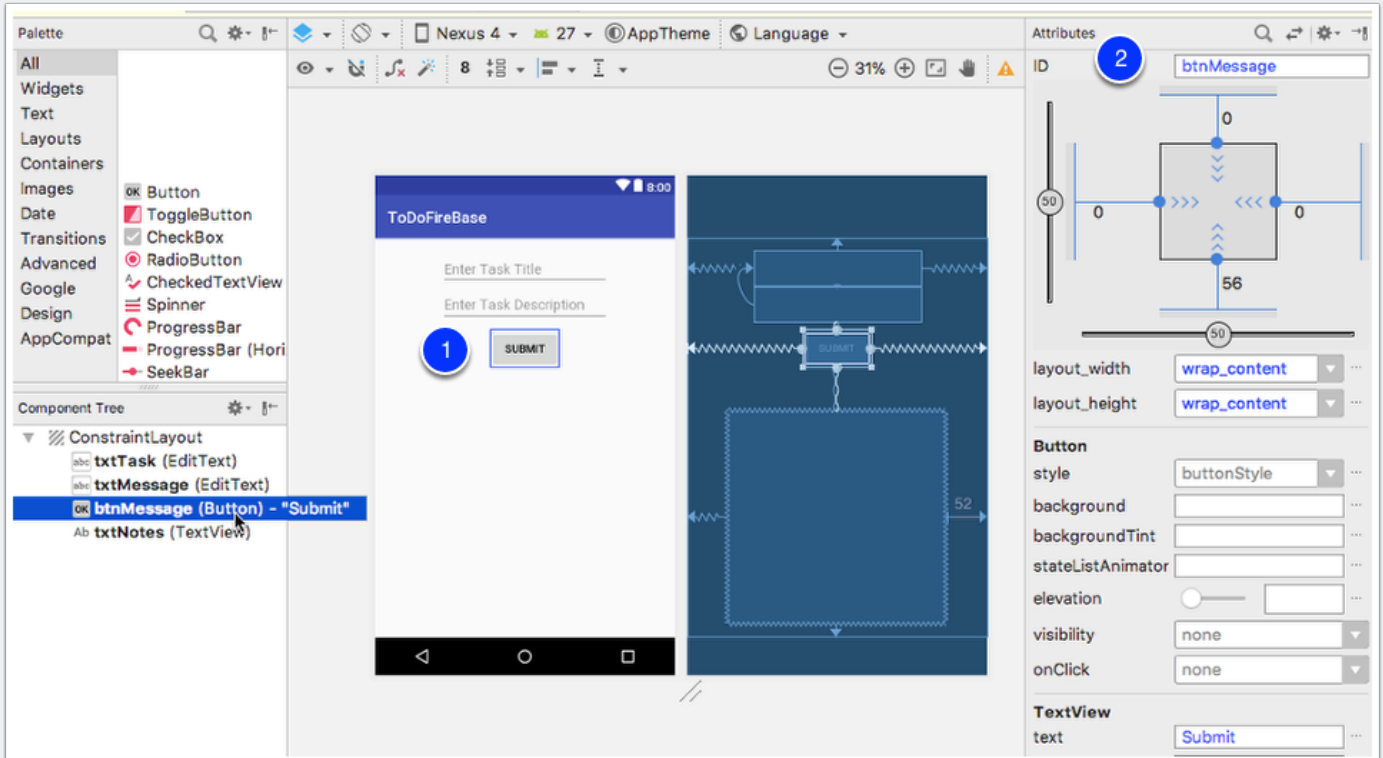
activity_main.xml - Add txtMessage

1. Add EditText
2. ID: txtMessage
3. Hint: Enter Task Description



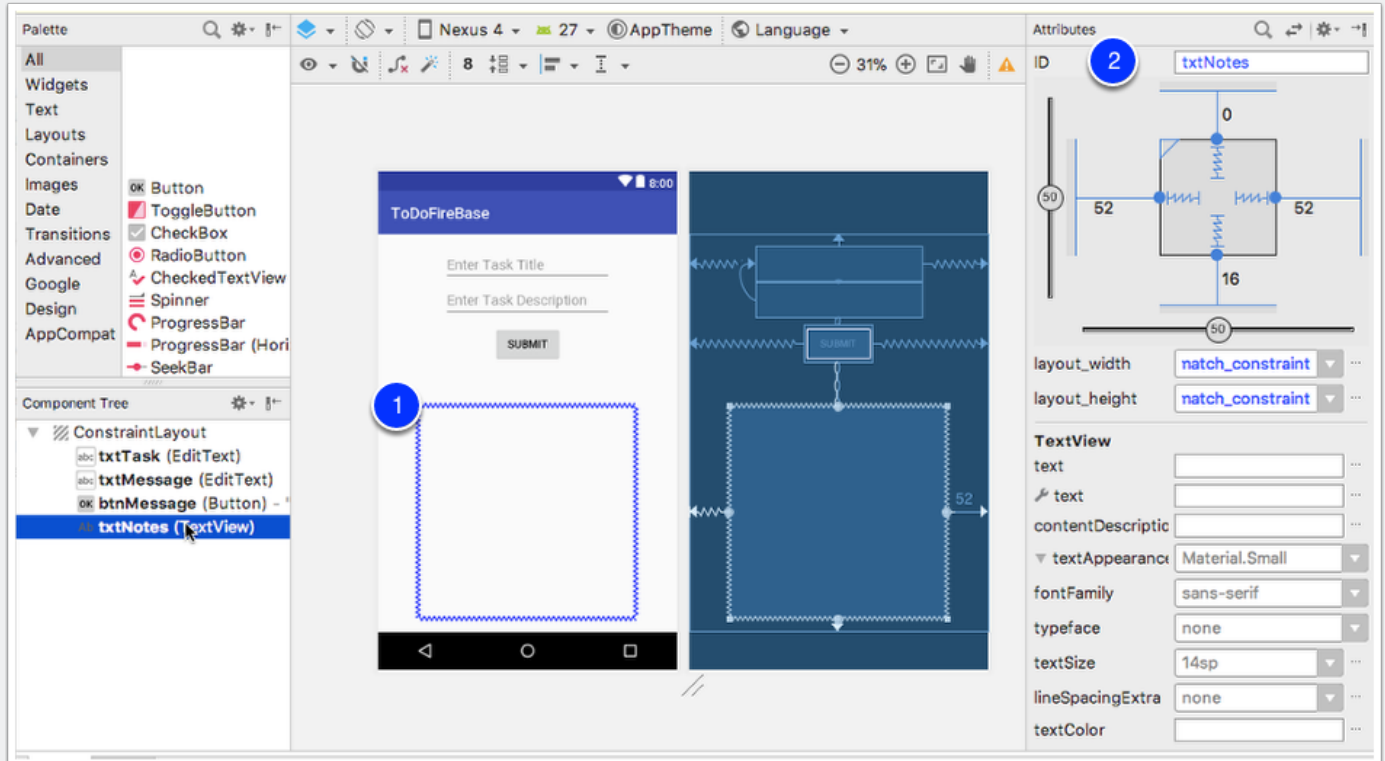
Adding btnMessage

1. Add Button
2. ID: btnMessage



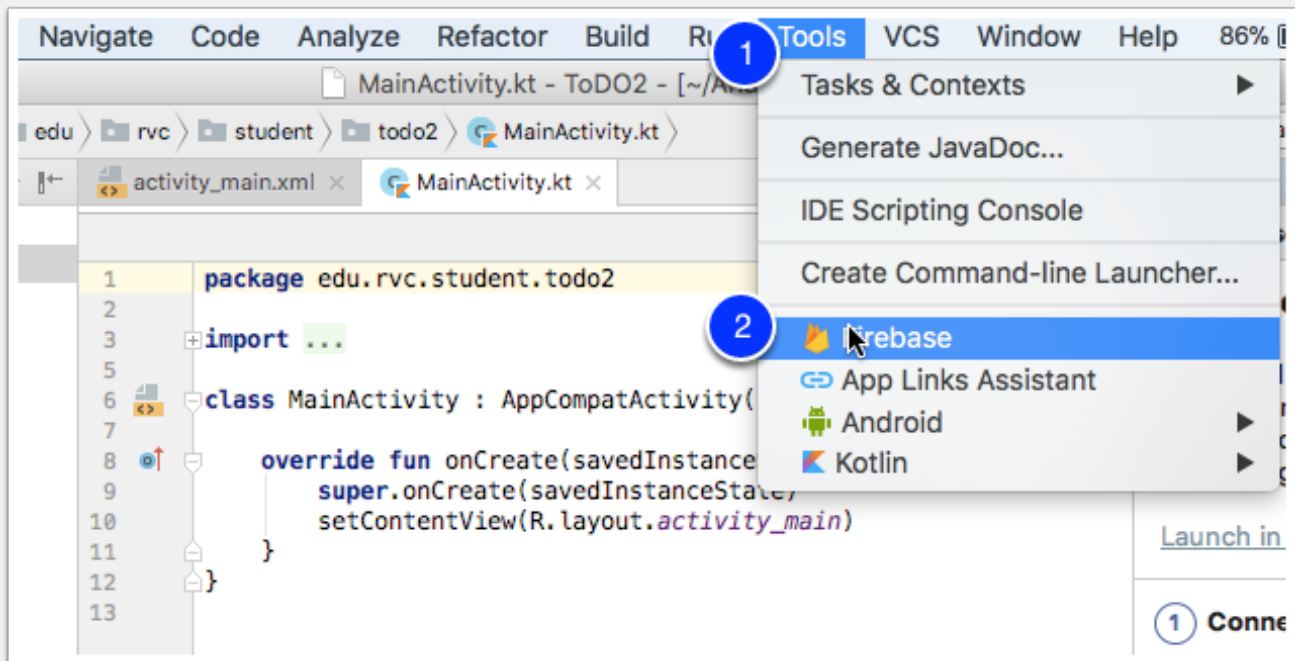
activity_main.xml - Add txtNotes

1. Add TextView
2. ID; txtNotes



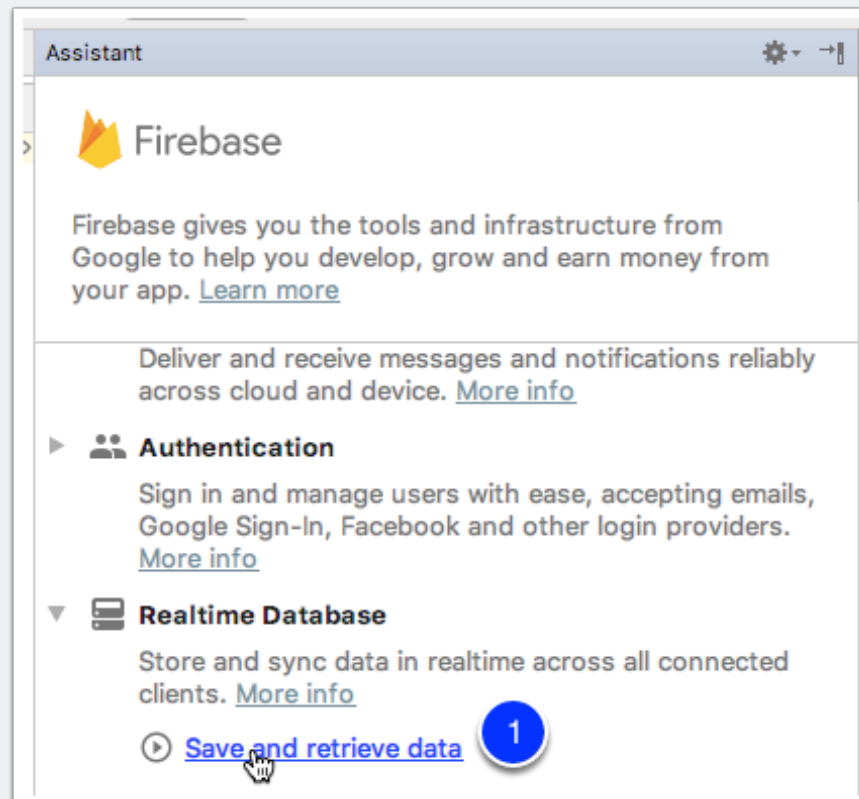
Create a FireBase Database Connection

1. Click on Tools
2. Click on Firecase



Create FireBase Connection

1. Click on Save and Retrieve data



Create FireBase Connection

Click on **Connect to Firebase**

[← Firebase](#) > Realtime Database

Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

[Launch in browser](#)

1

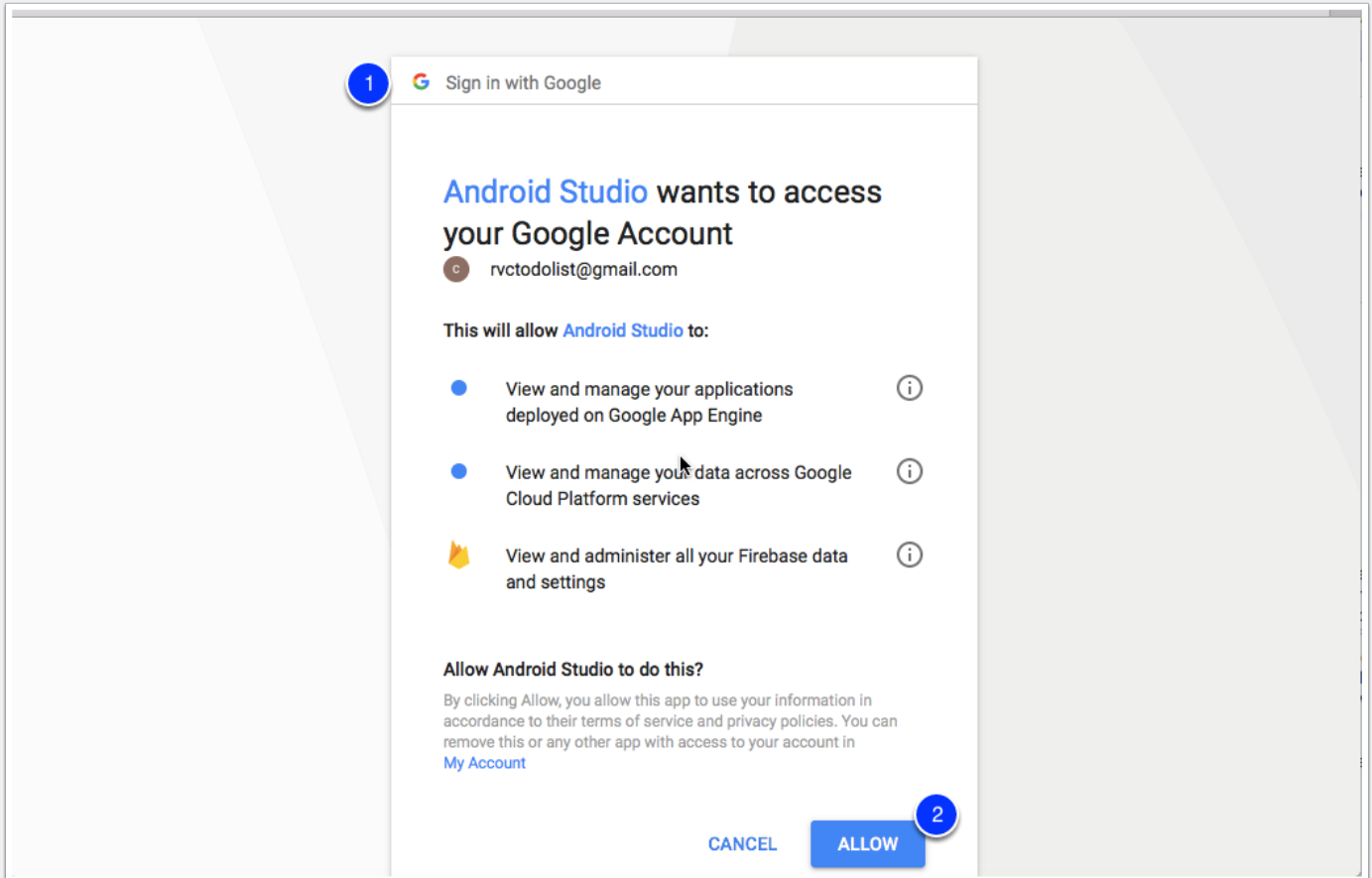
Connect your app to Firebase

Connect to Firebase

1

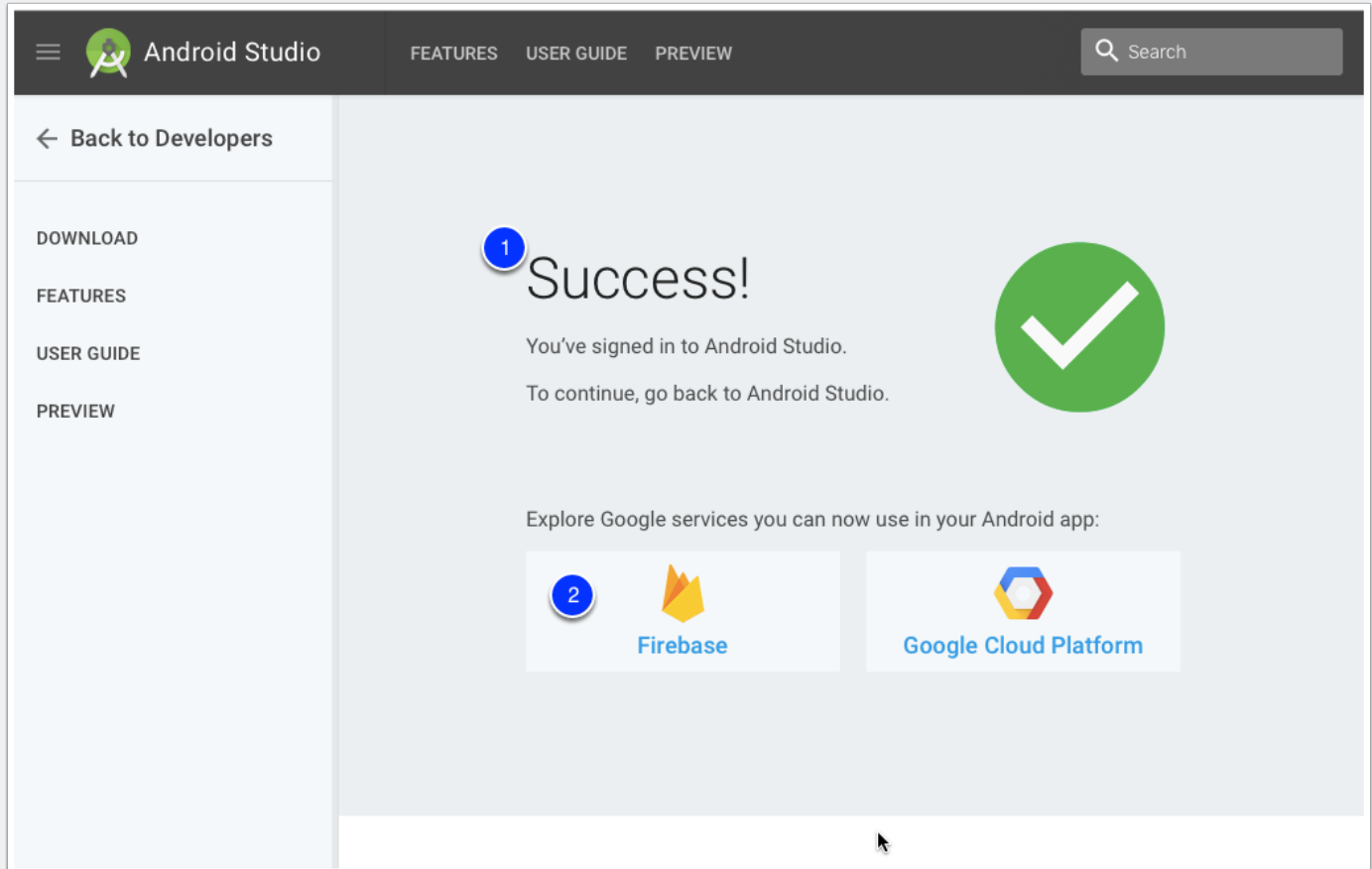
Sign in - Google Accounts

1. Sign-in to new Google account
2. Allow Android Studio to Access your new Google Account



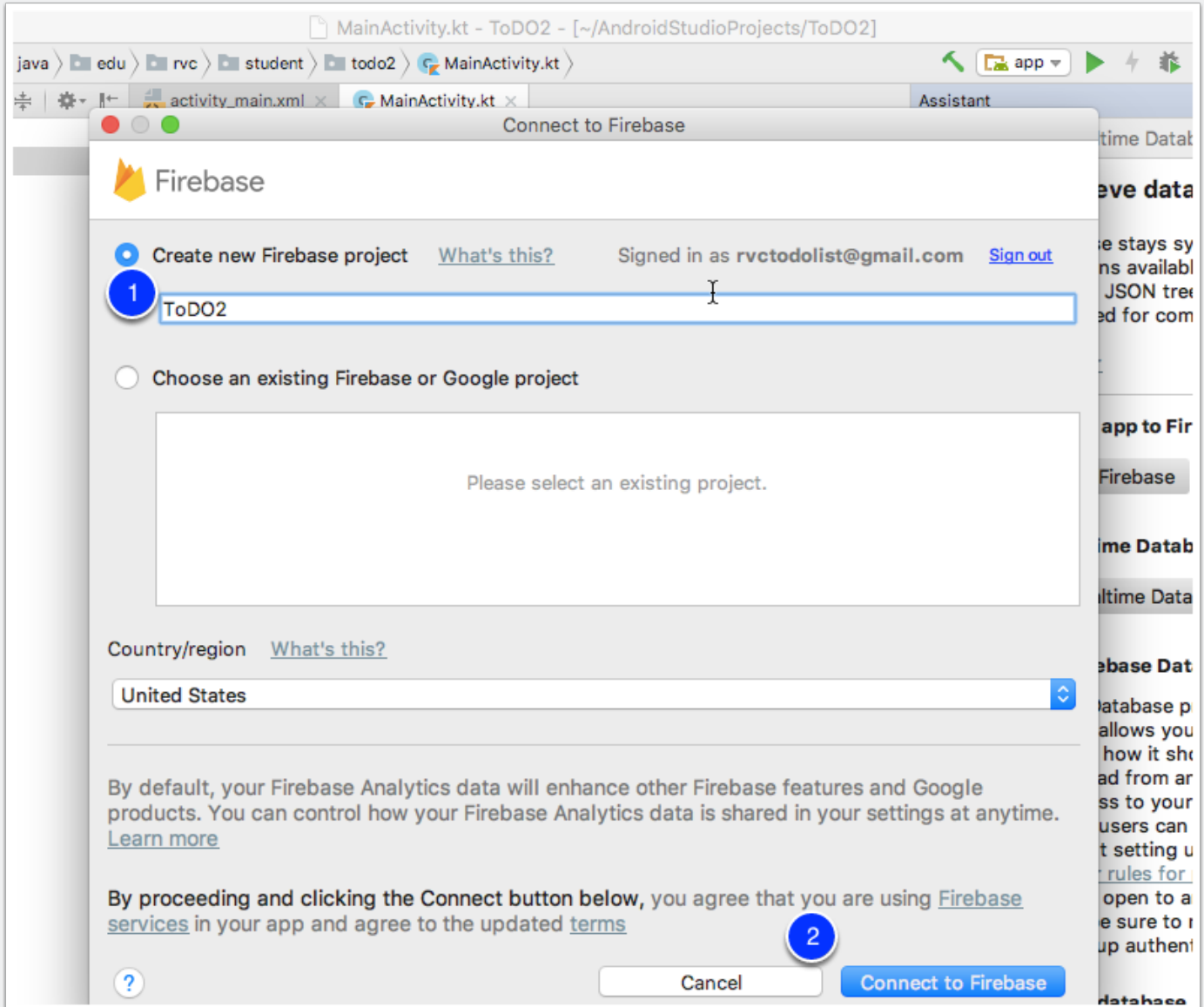
Sign into Android Studio | Android Studio

1. You Should See a Success Message!
2. Click on **Firestore**



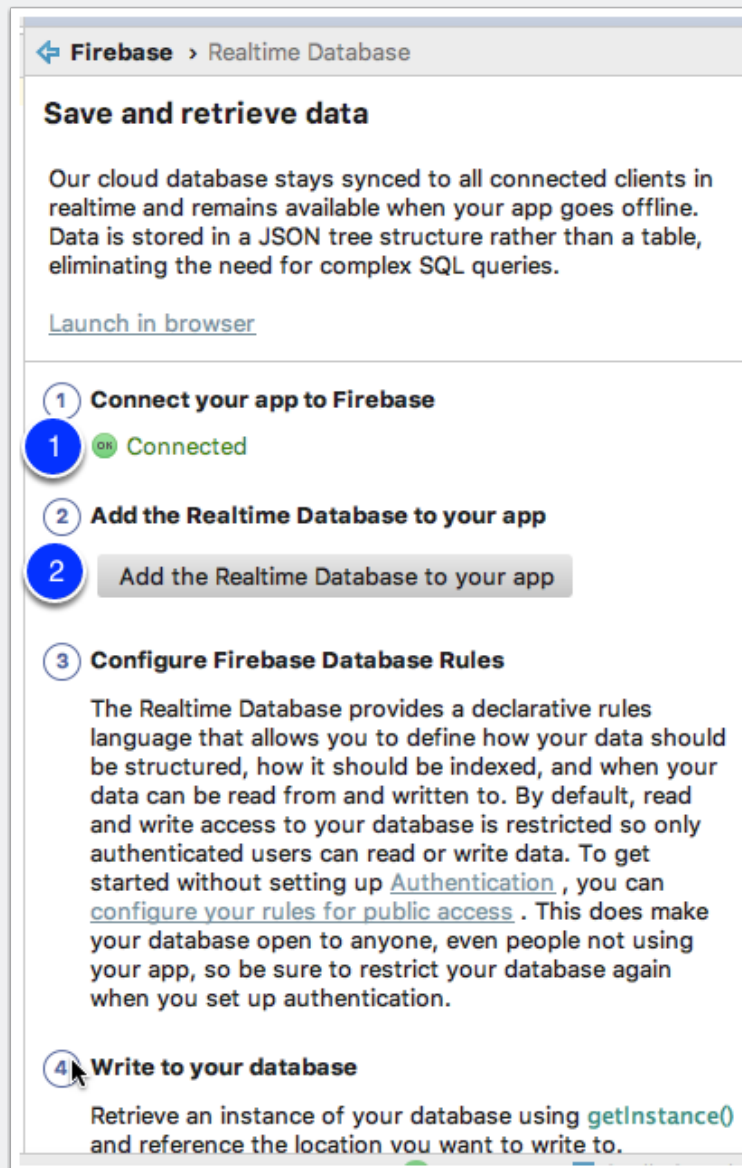
Connect to Firebase

1. It should populate with your new Android Project Name
2. Click **Connect to FireBase**



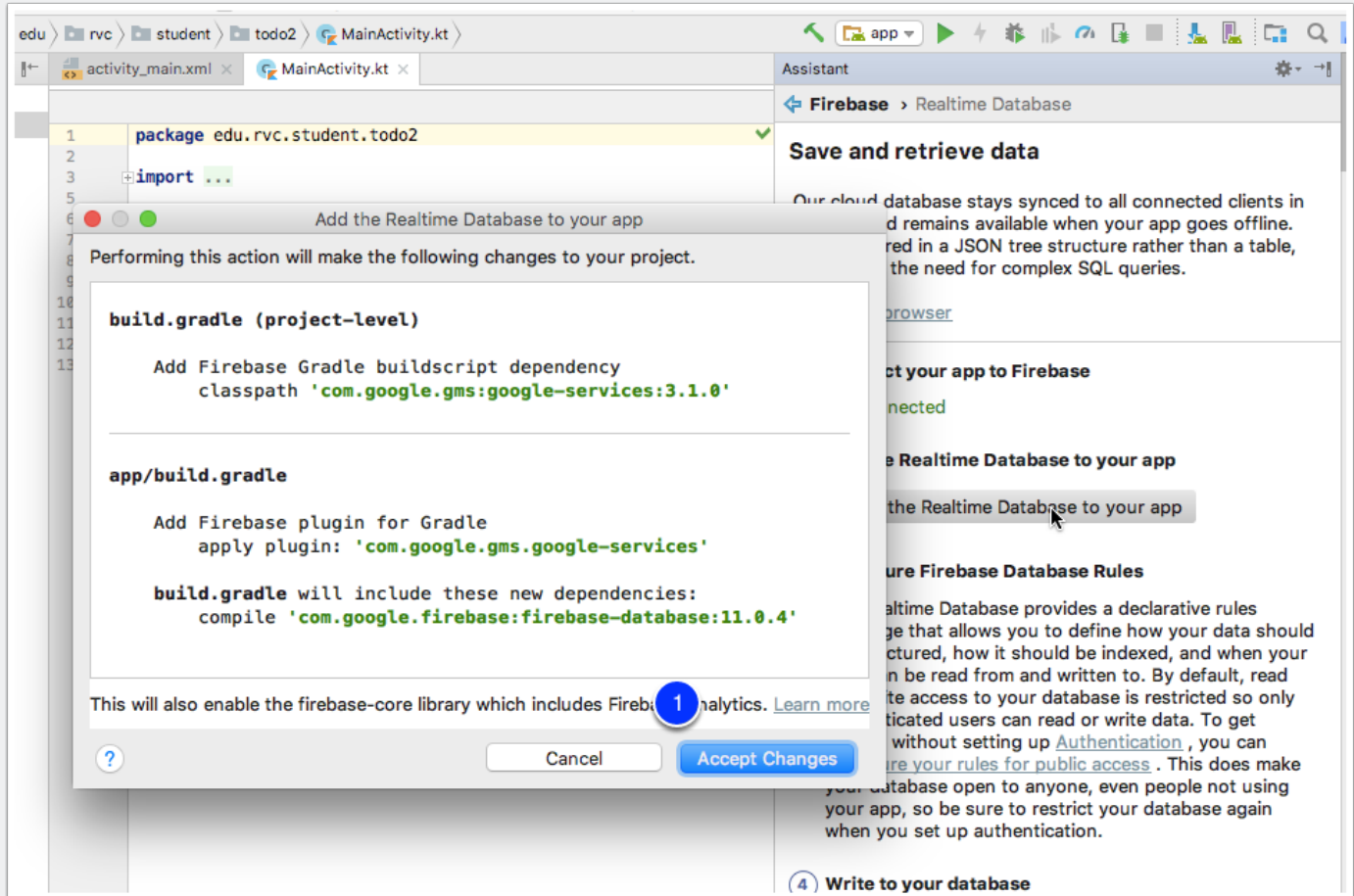
Connect to Firebase

1. Should see **Connected**
2. Click on **Add the Realtime Database to your app**



Add the Realtime Database to your app

1. Click **Accept Changes**
2. Allow to Update Gradle (takes a bit)



Connect to Firebase

Should see success (green icon) once Gradle is done

Save and retrieve data

Our cloud database stays synced to all connected clients in realtime and remains available when your app goes offline. Data is stored in a JSON tree structure rather than a table, eliminating the need for complex SQL queries.

[Launch in browser](#)

1

Connect your app to Firebase

OR

Connected

2

Add the Realtime Database to your app

1

Dependencies set up correctly

3

Configure Firebase Database Rules

The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up [Authentication](#), you can [configure your rules for public access](#). This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

4

Write to your database

Retrieve an instance of your database using `getInstance()` and reference the location you want to write to.

Connect to Firebase

1. Update Firebase Rules
2. Allow all to read and write
3. Click on **configure your rules for public access**

3 Configure Firebase Database Rules

The Realtime Database provides a declarative rules language that allows you to define how your data should be structured, how it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up [Authentication](#), you can [configure your rules for public access](#). This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

1

Connect to Firebase

Click on **GO TO CONSOLE**

The screenshot shows the Firebase documentation website. At the top, there's a navigation bar with the Firebase logo, links for Products, Use Cases, Pricing, and Documentation (which is highlighted). To the right of the navigation bar is a search bar and a 'GO TO CONSOLE' button with a notification badge. Below the navigation bar is a blue header with 'Documentation' and sub-links for OVERVIEW, GUIDES, REFERENCE, SAMPLES, and LIBRARIES. A yellow banner below the header announces 'Cloud Firestore (beta)'. The main content area features a sidebar on the left with a list of topics under 'Understand Rules', including 'Get Started' (which is highlighted). The main content area displays the title 'Get Started with Database Rules' with a five-star rating. Below the title is a table of contents with links to 'Contents', 'Configuring rules', 'Sample rules', and 'Next steps'. The introductory text states: 'The Firebase Realtime Database provides a flexible, expression-based rules language with JavaScript-like syntax to easily define how your data should be structured, how it should be'.

1 GO TO CONSOLE

Documentation

OVERVIEW GUIDES REFERENCE SAMPLES LIBRARIES SEND FEEDBACK

Announcing Cloud Firestore (beta): Try the new, scalable, flexible database from Firebase and Google Cloud Platform. [Learn more about Cloud Firestore.](#)

Understand Rules

- Get Started
- Secure Data
- Secure User Data
- Index Data
- Manage Rules via REST
- Usage and Performance
- Automated Backups
- Extend with Cloud Functions

Get Started with Database Rules

☆☆☆☆☆

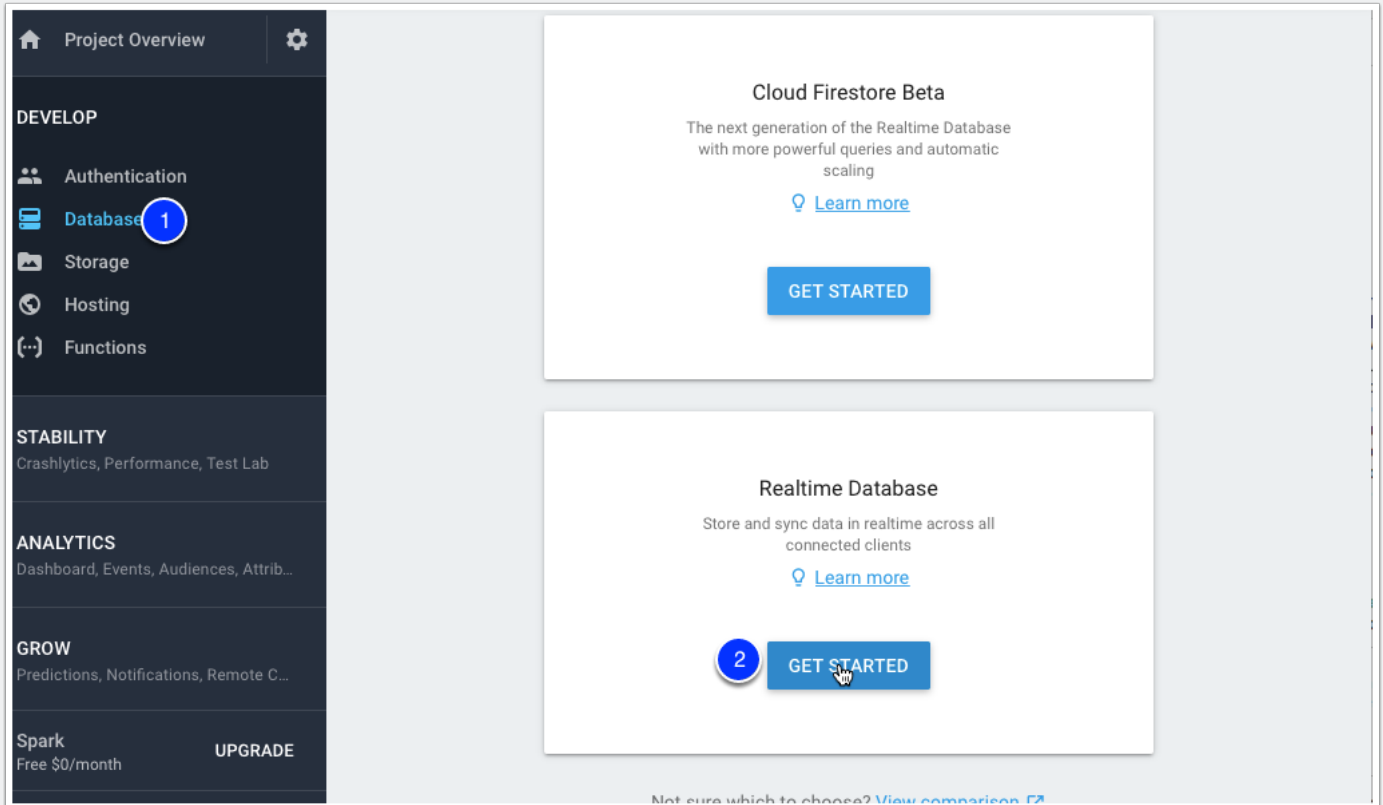
Contents

- Configuring rules
- Sample rules
- Next steps

The Firebase Realtime Database provides a flexible, expression-based rules language with JavaScript-like syntax to easily define how your data should be structured, how it should be

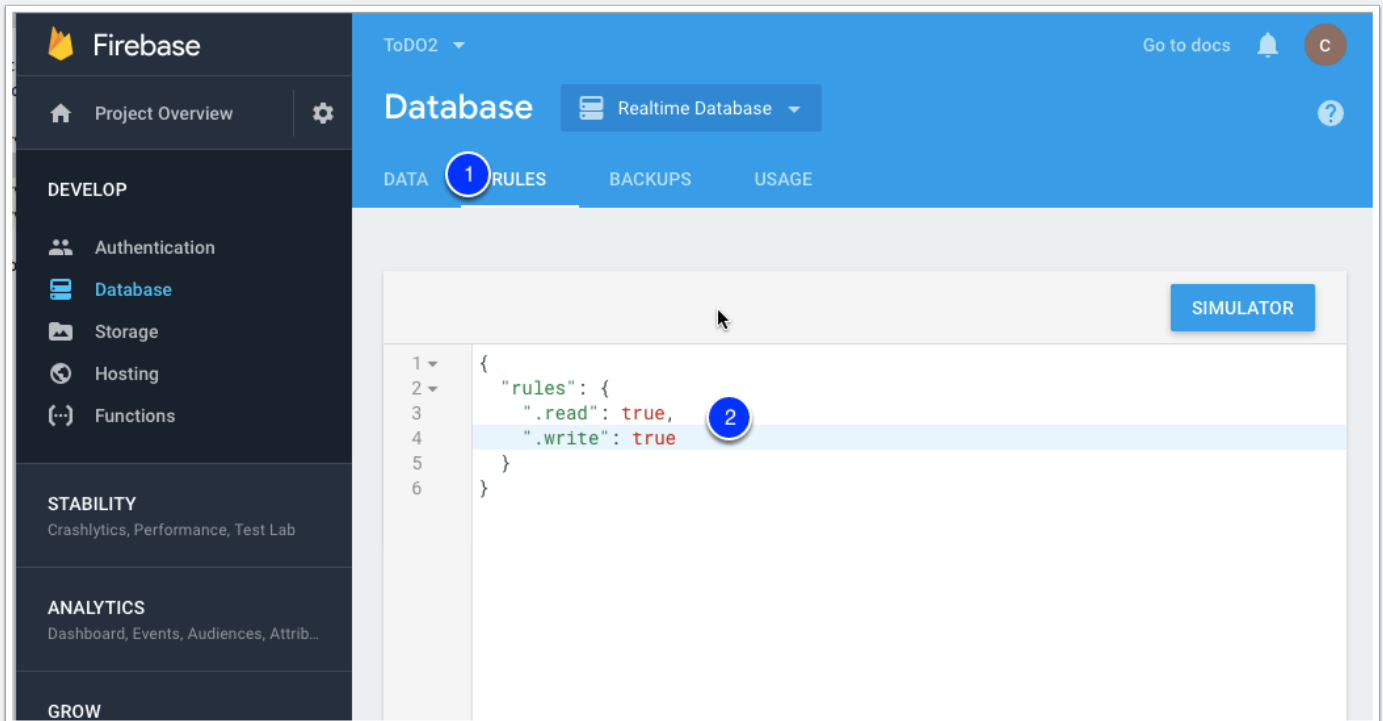
Connect to Firebase

1. Click on **Database**
2. Click on **Get Started** under Realtime Database



Connect to Firebase

1. Click on **RULES** tab
2. Make sure your rule looks like this
3. It allows all who use app to read and write to database



Connect to Firebase

1. Write a test message to database
2. Copy code under #4
3. Paste below line 15 in MainActivity.kt (It should convert to Kotlin)
4. Run App

```
1 package edu.rvc.student.todo2
2
3 import android.support.v7.app.AppCompatActivity
4 import android.os.Bundle
5 import com.google.firebase.database.DatabaseReference
6 import com.google.firebase.database.FirebaseDatabase
7
8
9
10 class MainActivity : AppCompatActivity() {
11
12     override fun onCreate(savedInstanceState: Bundle?) {
13         super.onCreate(savedInstanceState)
14         setContentView(R.layout.activity_main)
15
16         // Write a message to the database
17         val database = FirebaseDatabase.getInstance()
18         val myRef = database.getReference("message")
19
20         myRef.setValue("Hello, World!")
21     }
22 }
23
```

Firebase > Realtime Database

be structured, now it should be indexed, and when your data can be read from and written to. By default, read and write access to your database is restricted so only authenticated users can read or write data. To get started without setting up [Authentication](#), you can [configure your rules for public access](#). This does make your database open to anyone, even people not using your app, so be sure to restrict your database again when you set up authentication.

4 Write to your database

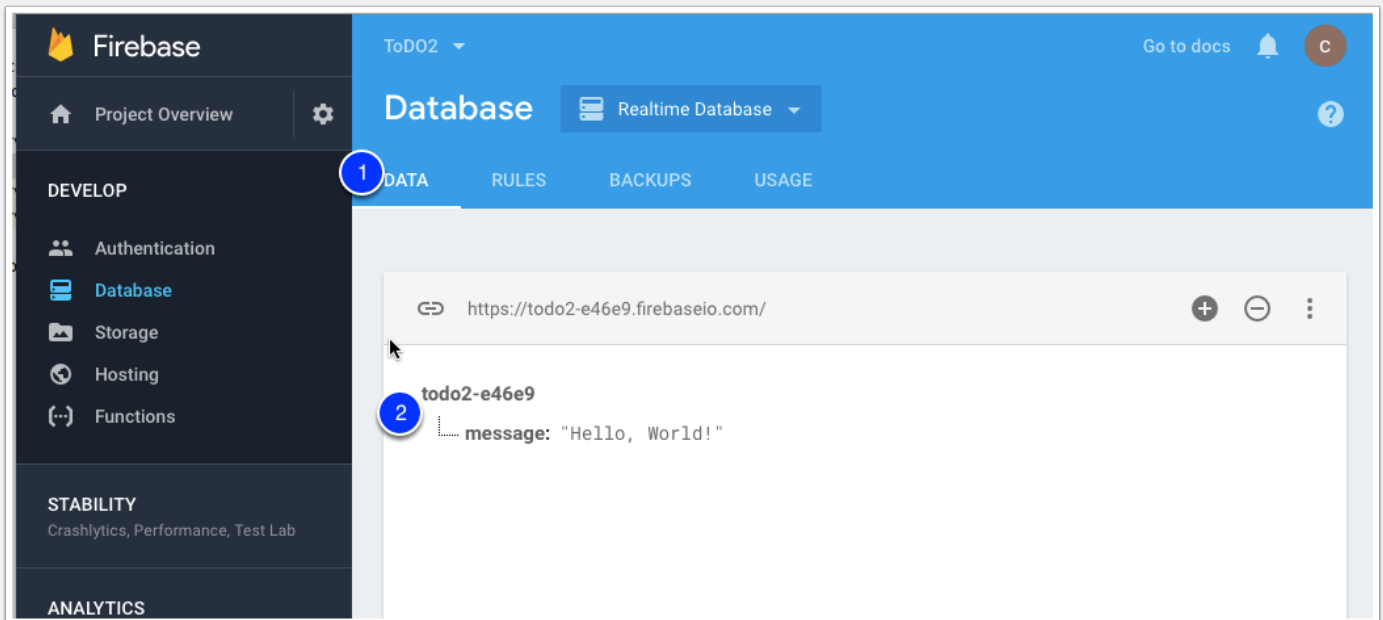
Retrieve an instance of your database using `getInstance()` and reference the location you want to write to.

```
// Write a message to the database
FirebaseDatabase database = FirebaseDatabase.getInstance()
DatabaseReference myRef = database.getReference("message")
myRef.setValue("Hello, World!");
```

You can save a range of data types to the database this way, including Java objects. When you save an object the responses from any getters will be saved as children of this location.

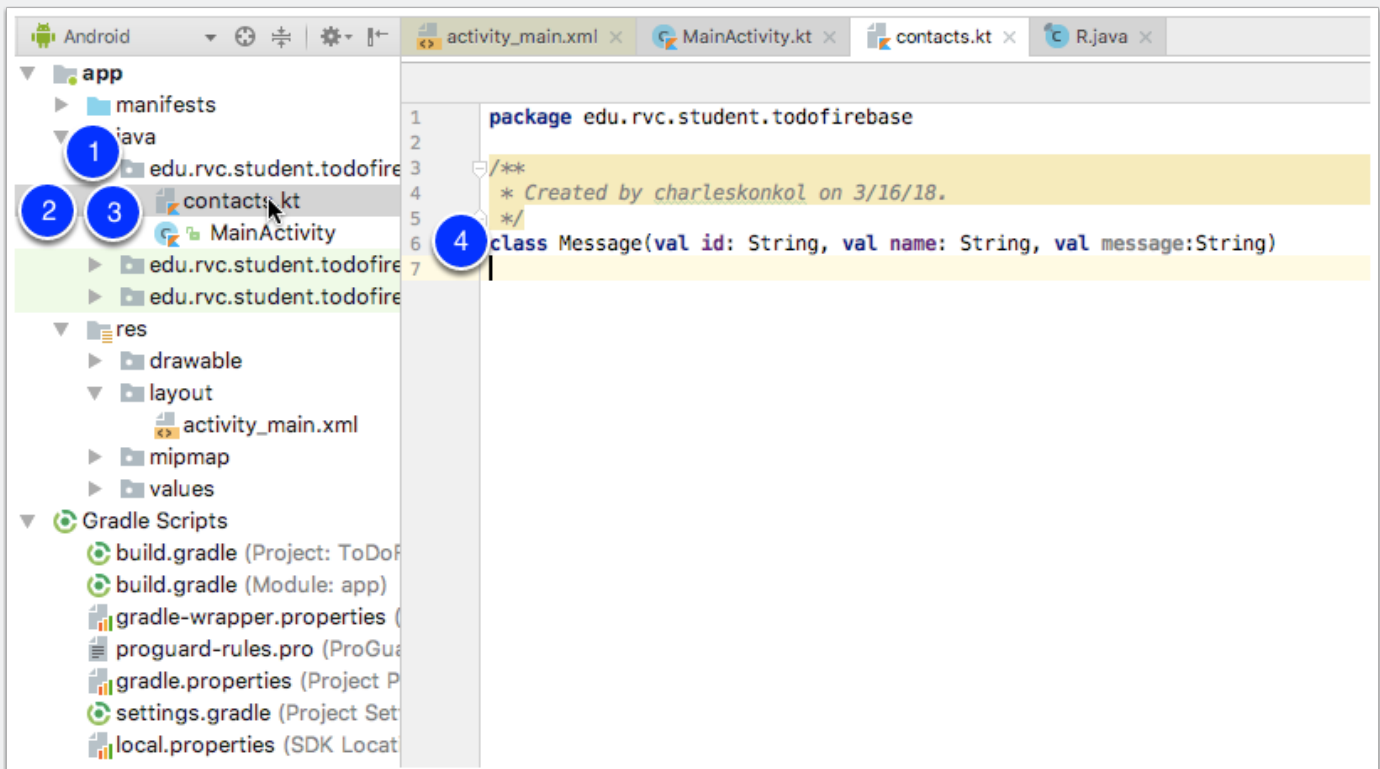
Connect to Firebase

1. Go Back to Firebase in your web browser
2. Click on **Data**
3. View **message: "Hello World!"** that was just written



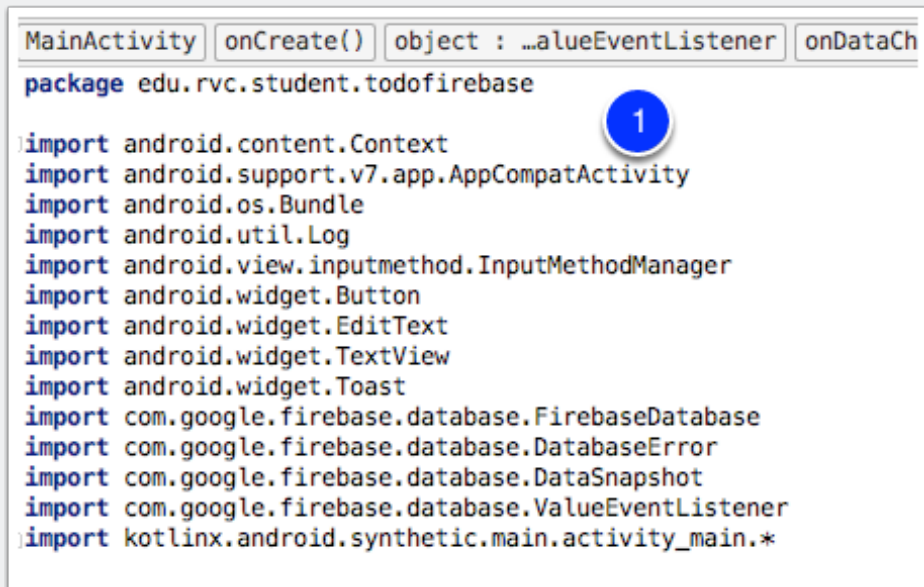
contacts.kt - Add file > contacts.kt

1. Right-Click on edu.rvc.stud... under java folder and select
2. New > Kotlin File Class
3. Add file > contacts.kt
4. Add Class Message on line 6 below



MainActivity.kt - Add Code

Make sure these imports are added to MainActivity.kt



```
MainActivity | onCreate() | object : ValueEventListener | onDataChangeCh
package edu.rvc.student.todofirebase

import android.content.Context
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.inputmethod.InputMethodManager
import android.widget.Button
import android.widget.EditText
import android.widget.TextView
import android.widget.Toast
import com.google.firebase.database.FirebaseDatabase
import com.google.firebase.database.DatabaseError
import com.google.firebase.database.DataSnapshot
import com.google.firebase.database.ValueEventListener
import kotlinx.android.synthetic.main.activity_main.*
```

MainActivity.kt - Add Code

1. Remove the test code earlier
2. Add Remaining Code



```
MainActivity | onCreate() | object : ...alueEventListener | onDataChange() | children.forEach{...} | if (mes

class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        1 var task = findViewById<EditText>(R.id.txtTask)
        2 var names = findViewById<EditText>(R.id.txtMessage)
        var btnMessage = findViewById<Button>(R.id.btnMessage)
        var messages = findViewById<TextView>(R.id.txtNotes)
        var ref = FirebaseDatabase.getInstance().getReference( p0: "Message")
        //var messages = findViewById<TextView>(R.id.txtNotes)
        //val messages = findViewById<TextView>(R.id.txtNotes)
        //firebase database
        //var ref = FirebaseDatabase.getInstance().getReference("Message")
        btnMessage.setOnClickListener{
            // Write a message to the database

            txtTask.requestFocus()

            var messageid = ref.push().key

            var messageg = Message(messageid, task.text.toString(), names.text.toString())

            hideKeyboard()
            task.setText("")
            names.setText("")
            txtTask.requestFocus()
            ref.child(messageid).setValue(messageg).addOnCompleteListener {
                Toast.makeText( context: this, text: "Task Added!", duration: 3).show()
            }
        }
    }
}
```

MainActivity.kt - Add Code

1. Add two functions **ref.addValueEventListener** & **hidekeyboard**

2. Test & confirm data is writing to database online
3. Submit to Github

```
1 // Listen and show data changes
ref.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        messages.text = ""
        val children = dataSnapshot.children
        children.forEach {
            println("data: " + it.toString())
            if (messages.text.toString() != "") {
                messages.text = messages.text.toString() + "\n" + "Task: " + it.child( p0: "name").value.toString() + " " + "Desc: " + it.child( p0: "message").value.toString()
            } else {
                messages.text = "My Tasks"
            }
            messages.text = messages.text.toString() + "\n" + "Task: " + it.child( p0: "name").value.toString() + " " + "Desc: " + it.child( p0: "message").value.toString()
        }
    }

    override fun onCancelled(error: DatabaseError) {
        // Failed to read value
        Log.w( tag: "Message", msg: "Failed to read value.", error.toException())
    }
})

// function to hide keyboard goes right before the last right bracket of Class MainActivity
//import android.content.Context
//import android.view.inputmethod.InputMethodManager
2 fun hideKeyboard() {
    try {
        val imm = getSystemService(Context.INPUT_METHOD_SERVICE) as InputMethodManager
        imm.hideSoftInputFromWindow(currentFocus!!.windowToken, flags: 0)
    } catch (e: Exception) {
        // TODO: handle exception
    }
}
```