Firebase

Core Functionality & Push Messaging

Jack Farmer & Maddie Holt
CPSC 575: Android Mobile Programming

Overview

1. What is Firebase
2. Why use Firebase
3. How does it work
4. How to get started
5. What you need
6. Demo

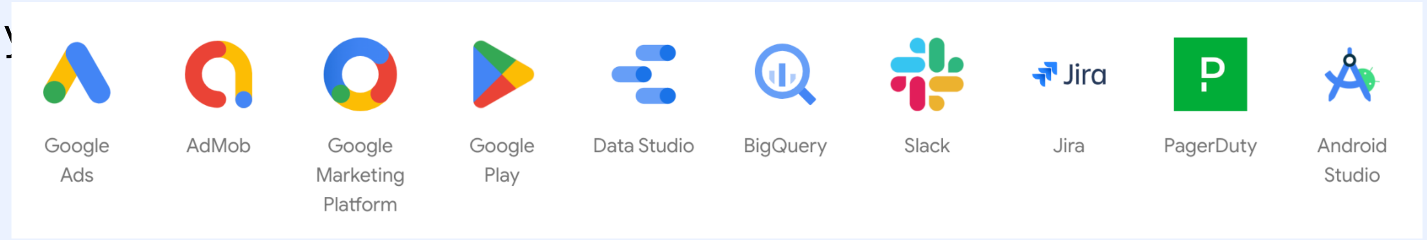
What is Firebase

- Google’s mobile and web app development platform
- Backend As A Service (BaaS)
- Products:

BUILD	**RELEASE & MONITOR**	**ENGAGE**
Accelerate and scale app development without managing infrastructure	Release with confidence and monitor performance and stability	Boost user engagement with rich analytics, A/B testing, and messaging campaigns

Why use Firebase

- Build, Release & Monitor, Engage products
- Firebase extensions
  - Install pre-packaged, open-source bundles of code to automate common development tasks
- Easily



- Easy to integrate on iOS, Android, and Web
  - Detailed documentation and cross-platform app development SDKs to help build and ship apps for iOS, Android, the web, Flutter, Unity, and C++

Why use Firebase?

- Competitors:
    - Microsoft Azure
    - Amazon Web Services
    - Airship
    - Ably
- Firebase Pros:
    - Much more affordable than competitors
    - Ease of implementation
    - Pre-managed backend services
- Firebase Cons:
    - Lack of global availability
    - Lesser iOS support than Android

How does it work?

- Easy to integrate into pre existing applications

```
FirebaseAuth auth = FirebaseAuth.getInstance();
auth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(new OnCompleteListener() {
        @Override
        public void onComplete(Task task) {
            if (task.isSuccessful()) {
                FirebaseUser user = task.getResult().getUser();
                String email = user.getEmail();
                // ...
            }
        }
    });
```

← authenticate a new user

subscribe to notification topic →

```
FirebaseMessaging.getInstance().subscribeToTopic("news");
```

How does it work?

Remote Logging

```
Bundle params = new Bundle();
params.putString("id", "image123");

FirebaseAnalytics.getInstance(this).logEvent("share_image", params);
```

```
FirebaseStorage storage = FirebaseStorage.getInstance();

storage.child("images/rivers.jpg").putBytes(data)
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception exception) {
            // ...
        }
    }).addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {
        @Override
        public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {
            // ...
        }
    });
```

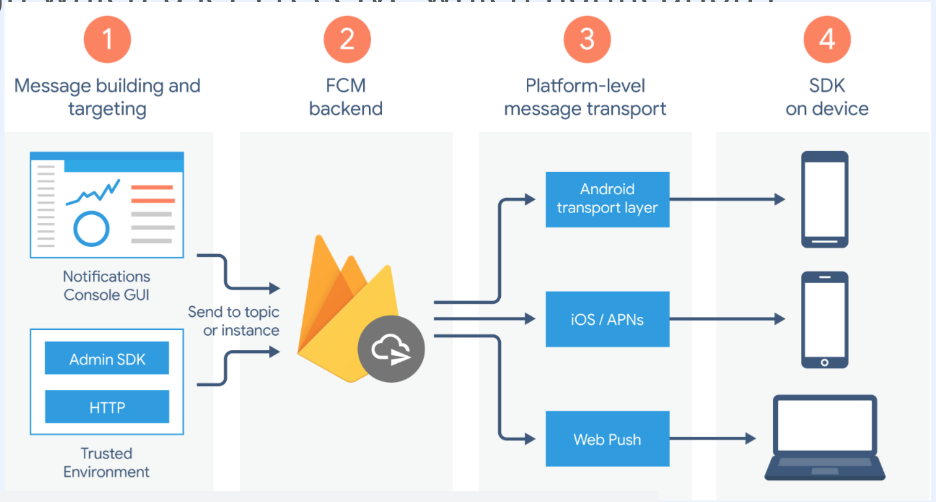
Saving files to remote destination

How does it work - Push notifications

- Centralized message delivery
- Firebase manages all notification “topics” that end users subscribe to
    - No need to manually sift through which users receive which notifications

Lifecycle

1. Message composed in trusted environment, sent to Firebase
2. Firebase receives message, generates metadata, sends to platform specific transport layer
3. Notification sent to end users when devices are online
4. End users receive notification


1
Message building and targeting
2
FCM backend
3
Platform-level message transport
4
SDK on device
Notifications Console GUI
Send to topic or instance
Admin SDK
HTTP
Trusted Environment
Android transport layer
iOS / APNs
Web Push

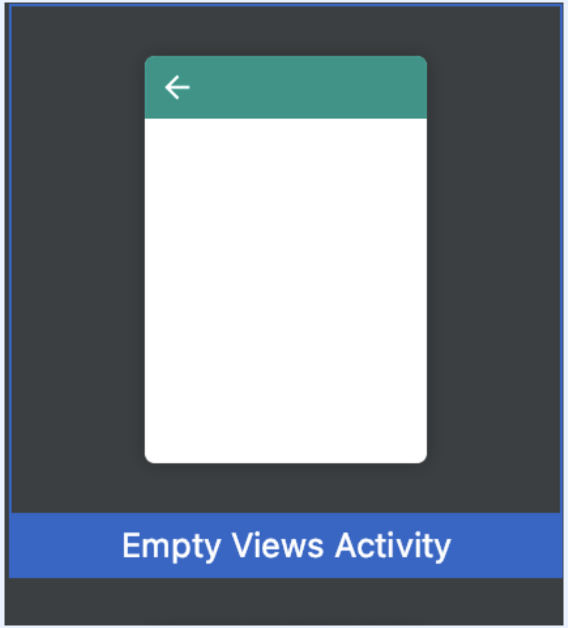
```
FirebaseMessaging.getInstance().subscribeToTopic("news");
```

Why does it work?

- Services backed by robust Google architecture
- See previous slides

What you need

- Empty View Activity / Pre-existing Android Project



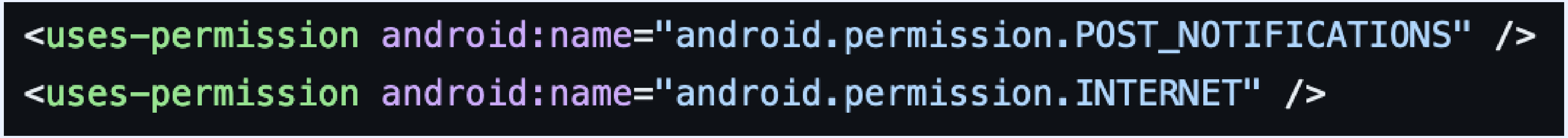
the easy part

What you need

- Grant Permission Access
  - Internet
  - Notifications

To send / receive notifications

Add in AndroidManifest.xml

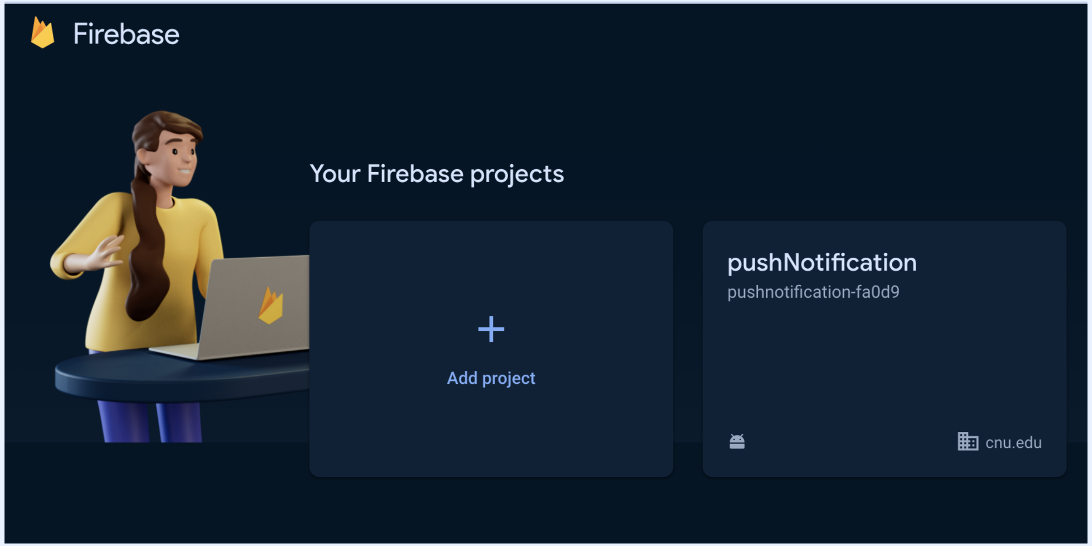


```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission android:name="android.permission.INTERNET" />
```

Internet

What you need

- Google Account
- Firebase Access



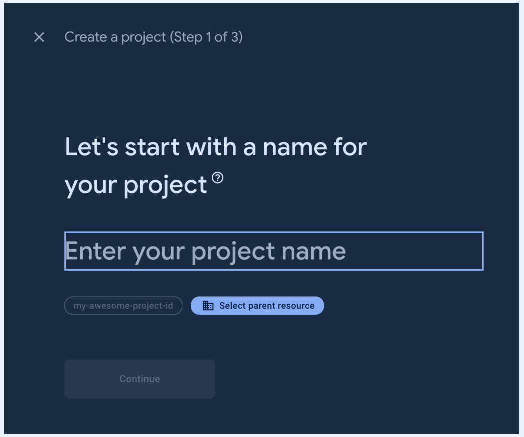
Your Firebase projects

pushNotification
pushnotification-fa0d9
cnu.edu

What you need

- Set up your Firebase project

1) Name your project



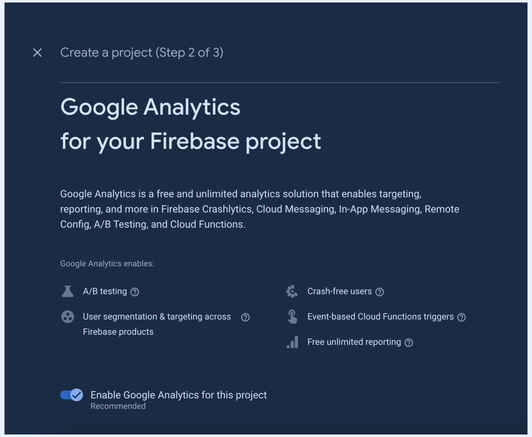
Create a project (Step 1 of 3)

Let's start with a name for your project

Enter your project name

my-awesome-project-id

2) Enable Google Analytics?



Create a project (Step 2 of 3)

Google Analytics for your Firebase project

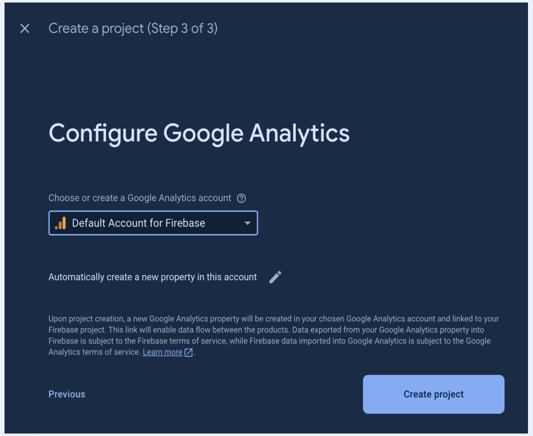
Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

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

3) Configure Google Analytics



Create a project (Step 3 of 3)

Configure Google Analytics

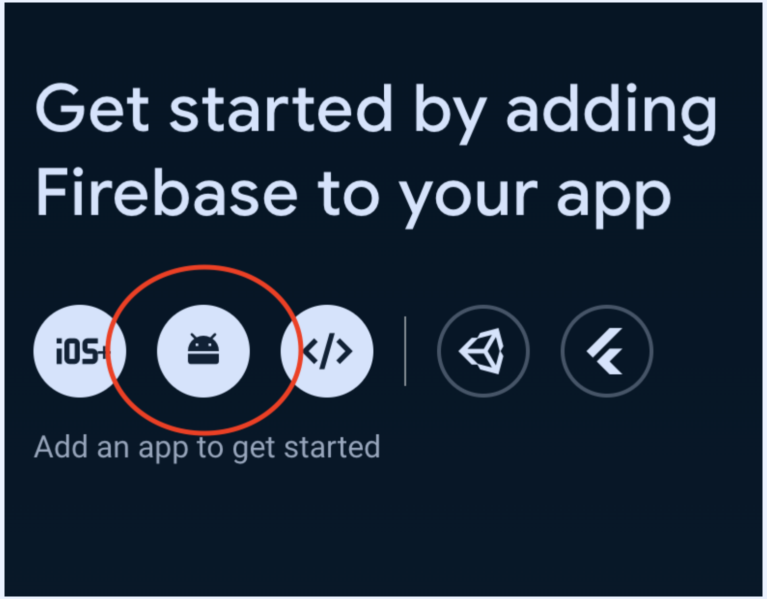
Choose or create a Google Analytics account: Default Account for Firebase

Automatically create a new property in this account

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more.

What you need

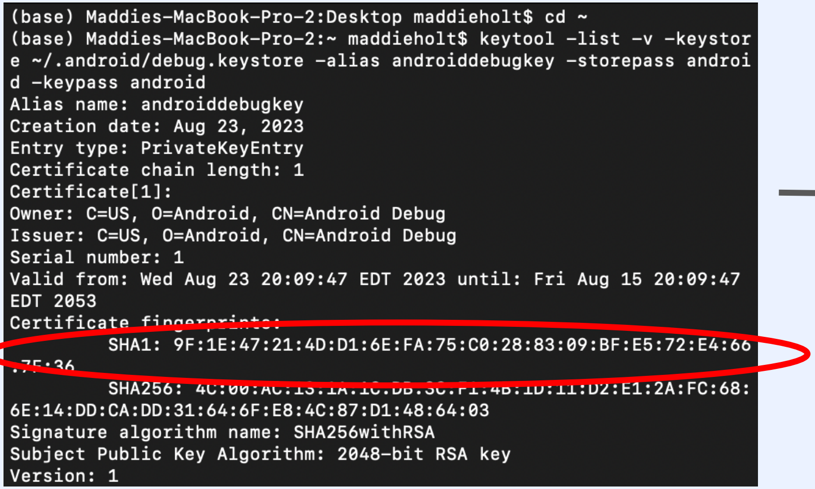
- Register your app with Firebase
  - (i.e. add your app to your Firebase project)
- Proceed with prompts


Get started by adding Firebase to your app

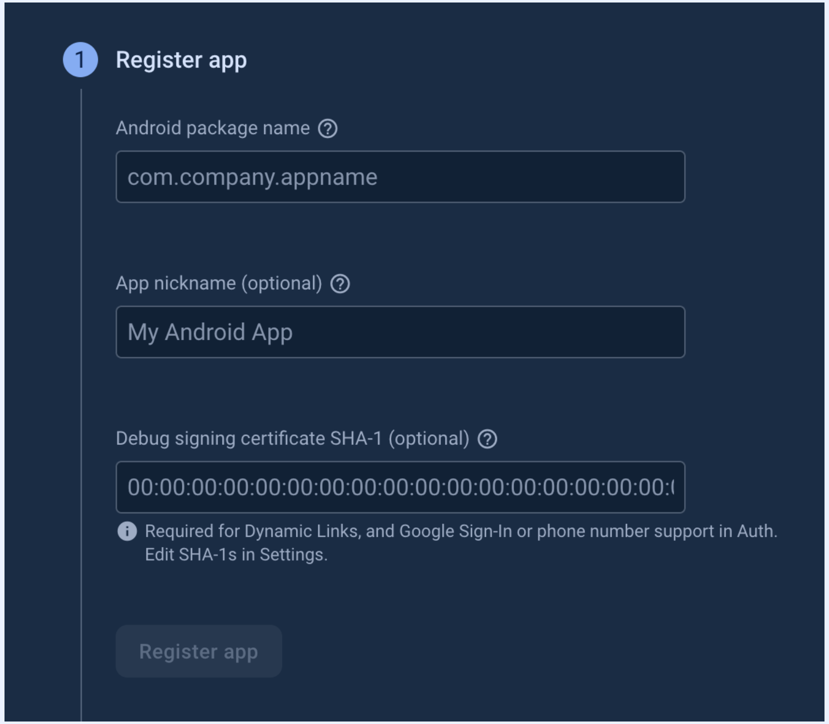
Add an app to get started

What you need

Ex: com.example.pushnotification



```
(base) Maddies-MacBook-Pro-2:Desktop maddieholt$ cd ~
(base) Maddies-MacBook-Pro-2:~ maddieholt$ keytool -list -v -keystor
e ~/.android/debug.keystore -alias androiddebugkey -storepass androi
d -keypass android
Alias name: androiddebugkey
Creation date: Aug 23, 2023
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: C=US, O=Android, CN=Android Debug
Issuer: C=US, O=Android, CN=Android Debug
Serial number: 1
Valid from: Wed Aug 23 20:09:47 EDT 2023 until: Fri Aug 15 20:09:47
EDT 2053
Certificate fingerprints:
         SHA1: 9F:1E:47:21:4D:D1:6E:FA:75:C0:28:83:09:BF:E5:72:E4:66
:7F:36
         SHA256: 4C:00:AC:13:1A:1C:DB:3C:F1:4B:1D:11:D2:E1:2A:FC:68:
6E:14:DD:CA:DD:31:64:6F:E8:4C:87:D1:48:64:03
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 1
```



1 Register app

Android package name

com.company.appname

App nickname (optional)

My Android App

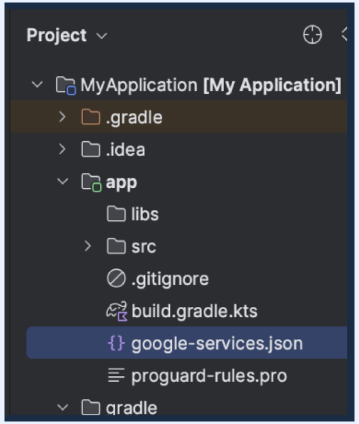
Debug signing certificate SHA-1 (optional)

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

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

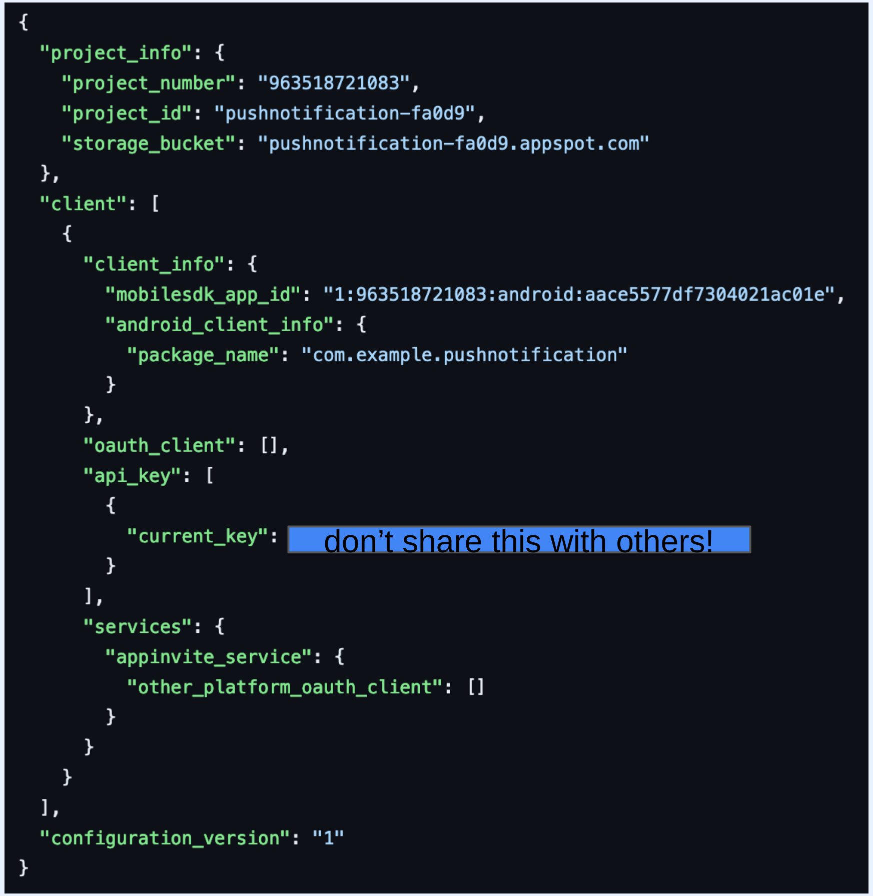
What you need

- Add Firebase Google-Services.json to app directory



```
Project ⌄
⌄ MyApplication [My Application]
  › .gradle
  › .idea
  ⌄ app
      libs
    › src
      .gitignore
      build.gradle.kts
      {} google-services.json
      proguard-rules.pro
  ⌄ gradle
```

google-services.json →

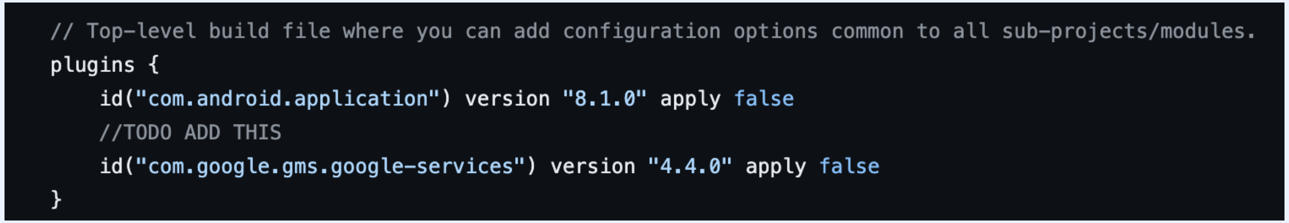


```
{
  "project_info": {
    "project_number": "963518721083",
    "project_id": "pushnotification-fa0d9",
    "storage_bucket": "pushnotification-fa0d9.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:963518721083:android:aace5577df7304021ac01e",
        "android_client_info": {
          "package_name": "com.example.pushnotification"
        }
      },
      "oauth_client": [],
      "api_key": [
        {
          "current_key": don't share this with others!
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_client": []
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

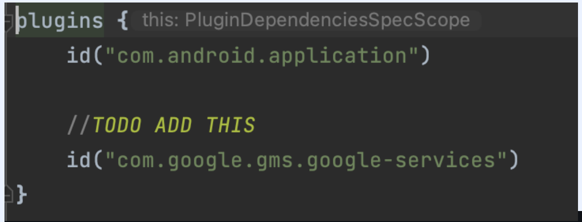
What you need

- To make the google-services.json config values accessible to Firebase SDKs, you need the Google services Gradle plugin

Project level $\longrightarrow$

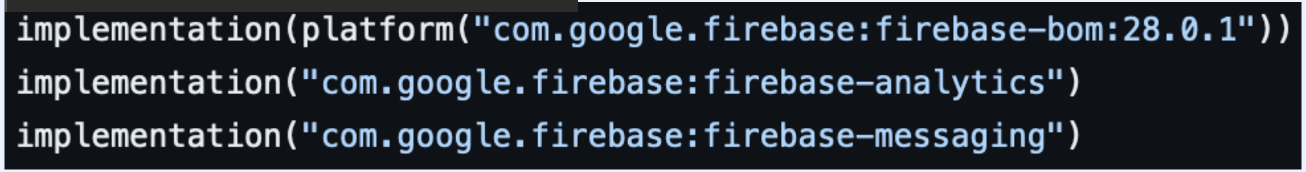


```
// Top-level build file where you can add configuration options common to all sub-projects/modules.
plugins {
    id("com.android.application") version "8.1.0" apply false
    //TODO ADD THIS
    id("com.google.gms.google-services") version "4.4.0" apply false
}
```



```
plugins {  this: PluginDependenciesSpecScope
    id("com.android.application")

    //TODO ADD THIS
    id("com.google.gms.google-services")
}
```



```
implementation(platform("com.google.firebase:firebase-bom:28.0.1"))
implementation("com.google.firebase:firebase-analytics")
implementation("com.google.firebase:firebase-messaging")
```

app level

What you need

- Add additional classes to App project that extend Firebase functionality and register class as service in AndroidManifest.xml
  - Ex: Push Notifications

```java
public class PushNotificationService extends FirebaseMessagingService
```

Service class

AndroidManifest.xml

```xml
<service android:name=".PushNotificationService" android:exported="false">
    <intent-filter>
        <action android:name = "com.google.firebase.MESSAGING_EVENT"></action>
    </intent-filter>
</service>
```

Demo

- Let’s put it all together & see it in action