Enabling Google Maps in your Android Project

From time to time Google changes the way it does things, and old tutorials may not apply to some new procedures.

This is another tutorial which, in about 6 months, will probably be irrelevant.

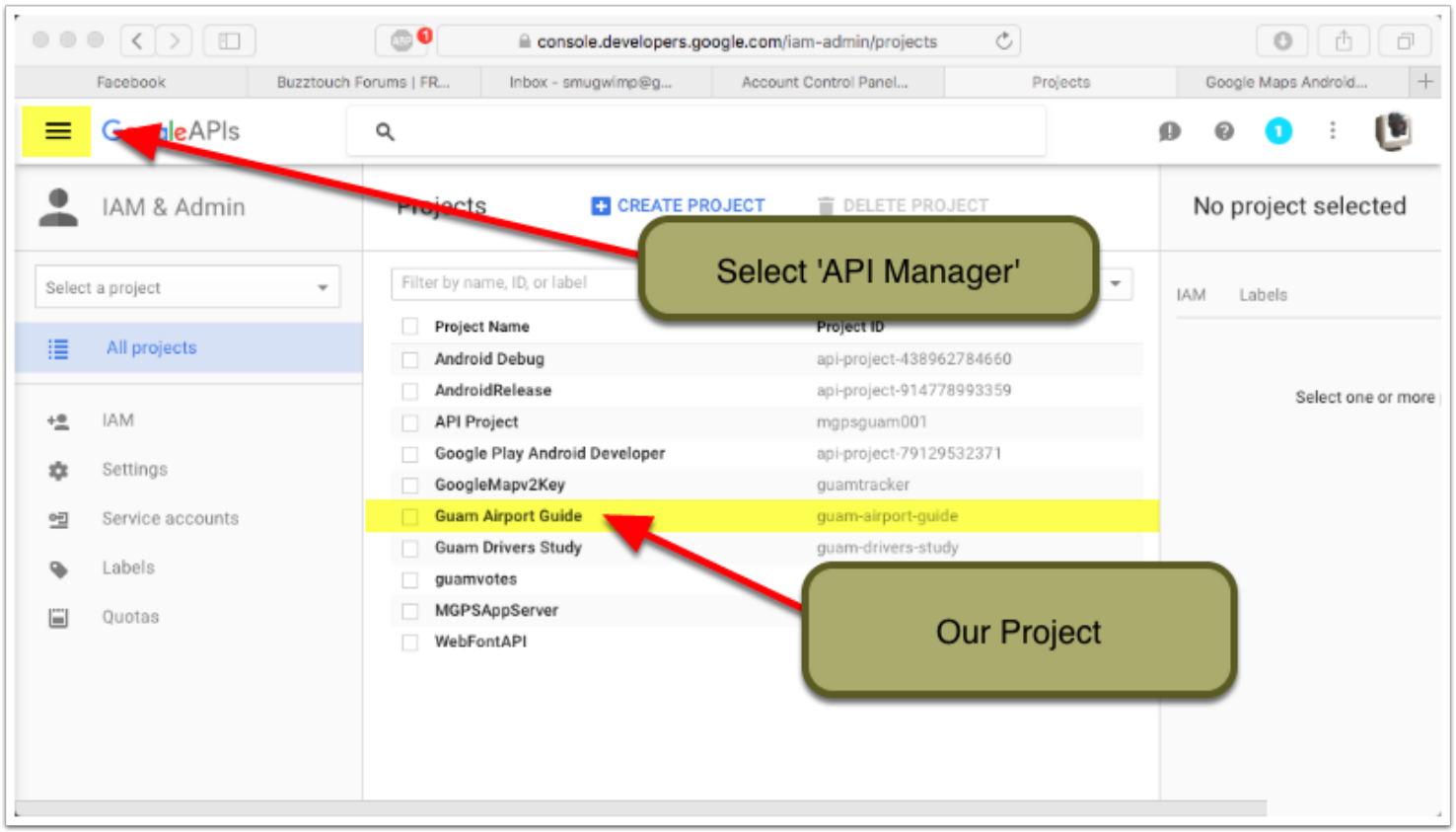
But until then...

Login to your Google Developer Console

You need to know how to get at least this far.

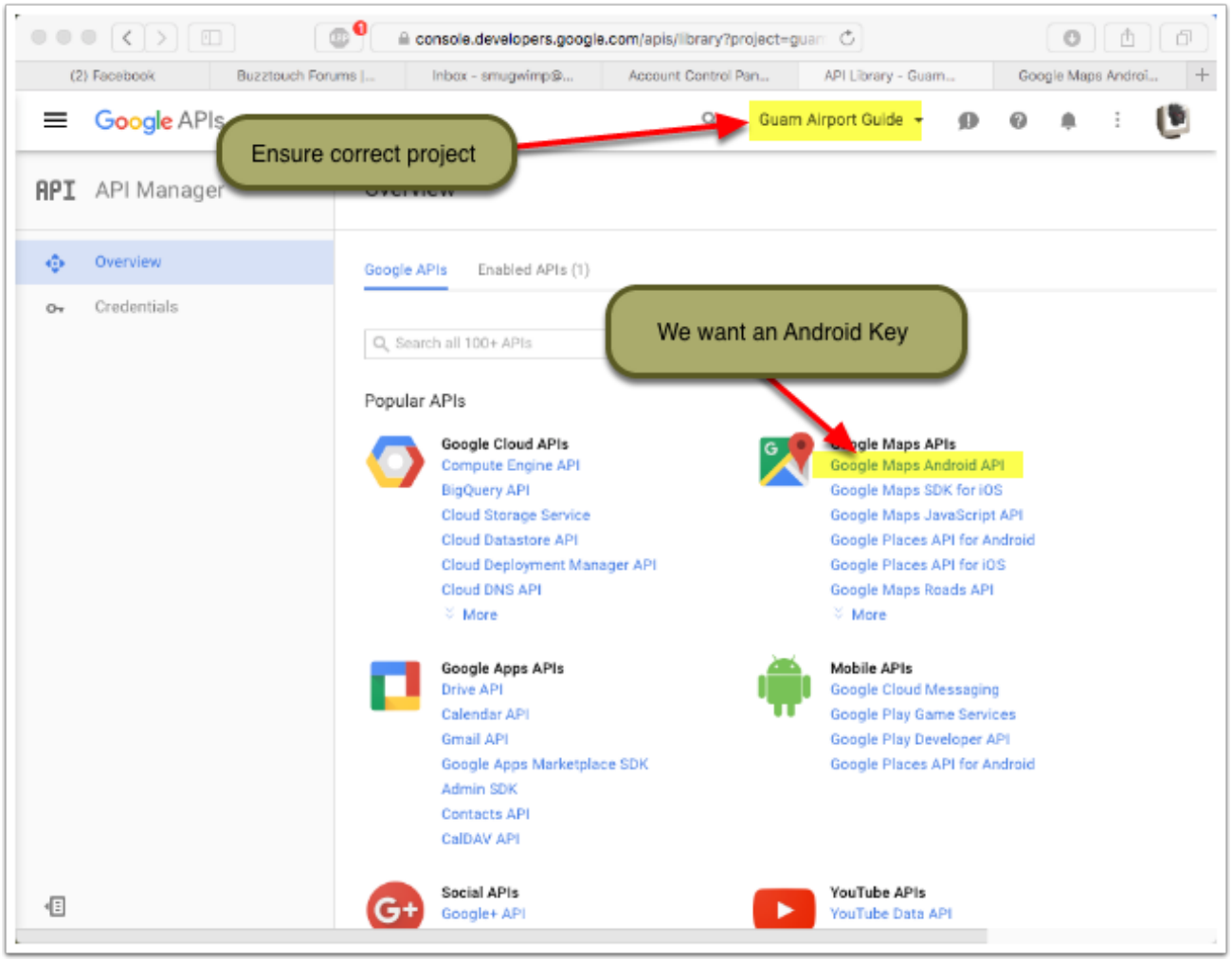
Login to your Google Developer Console

Create a project, or select a project to obtain an API



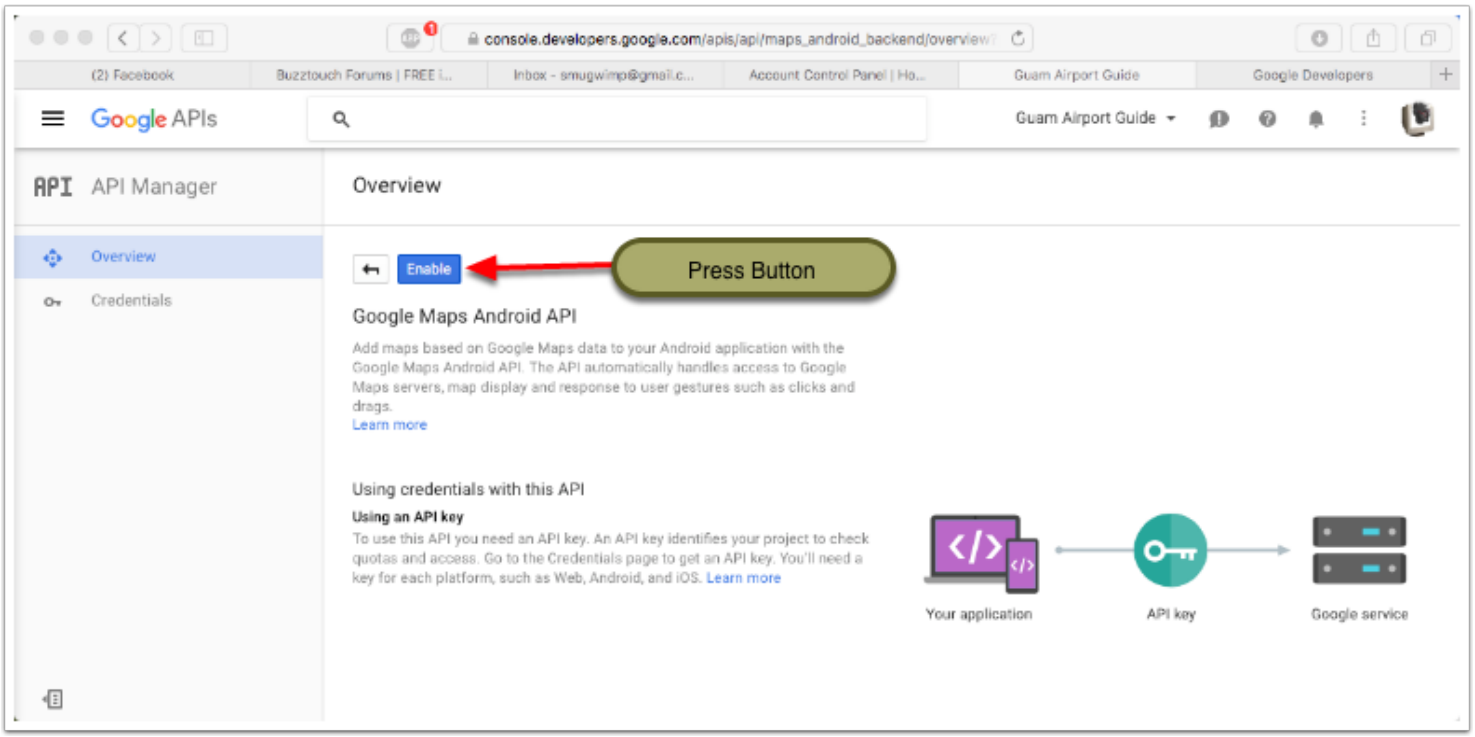
Select 'API Manager' for your project

Click on 'Google Maps Android API'

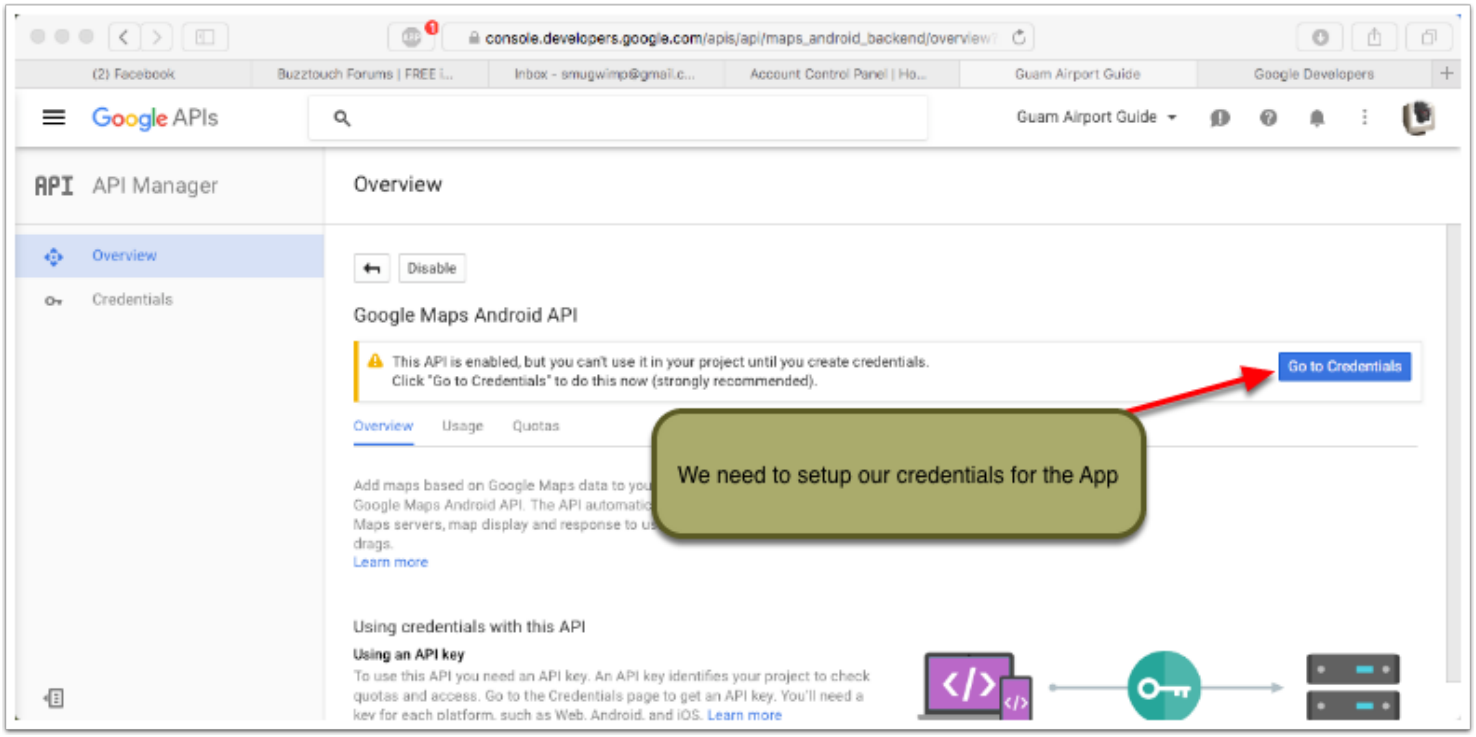


Enable the API

Press the 'Enable' button



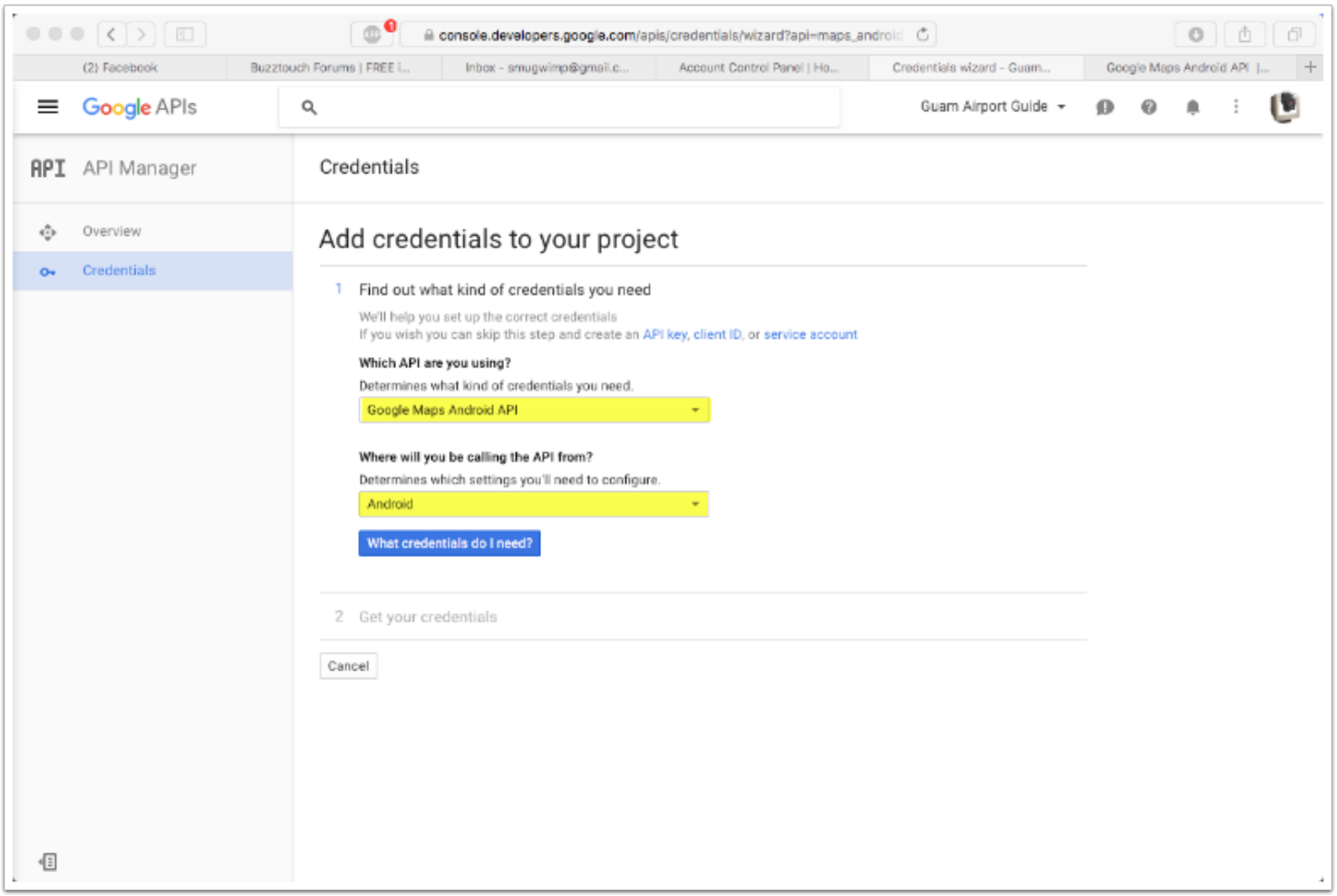
Enabled, but...


console.developers.google.com/apis/api/maps_android_backend/overview
Google APIs
Guam Airport Guide
API Manager
Overview
Credentials
Disable
Google Maps Android API
This API is enabled, but you can't use it in your project until you create credentials.
Click "Go to Credentials" to do this now (strongly recommended).
Go to Credentials
Overview
Usage
Quotas
We need to setup our credentials for the App
Learn more
Using credentials with this API
Using an API key
To use this API you need an API key. An API key identifies your project to check quotas and access. Go to the Credentials page to get an API key. You'll need a key for each platform, such as Web, Android, and iOS. Learn more

Select your API Parameters

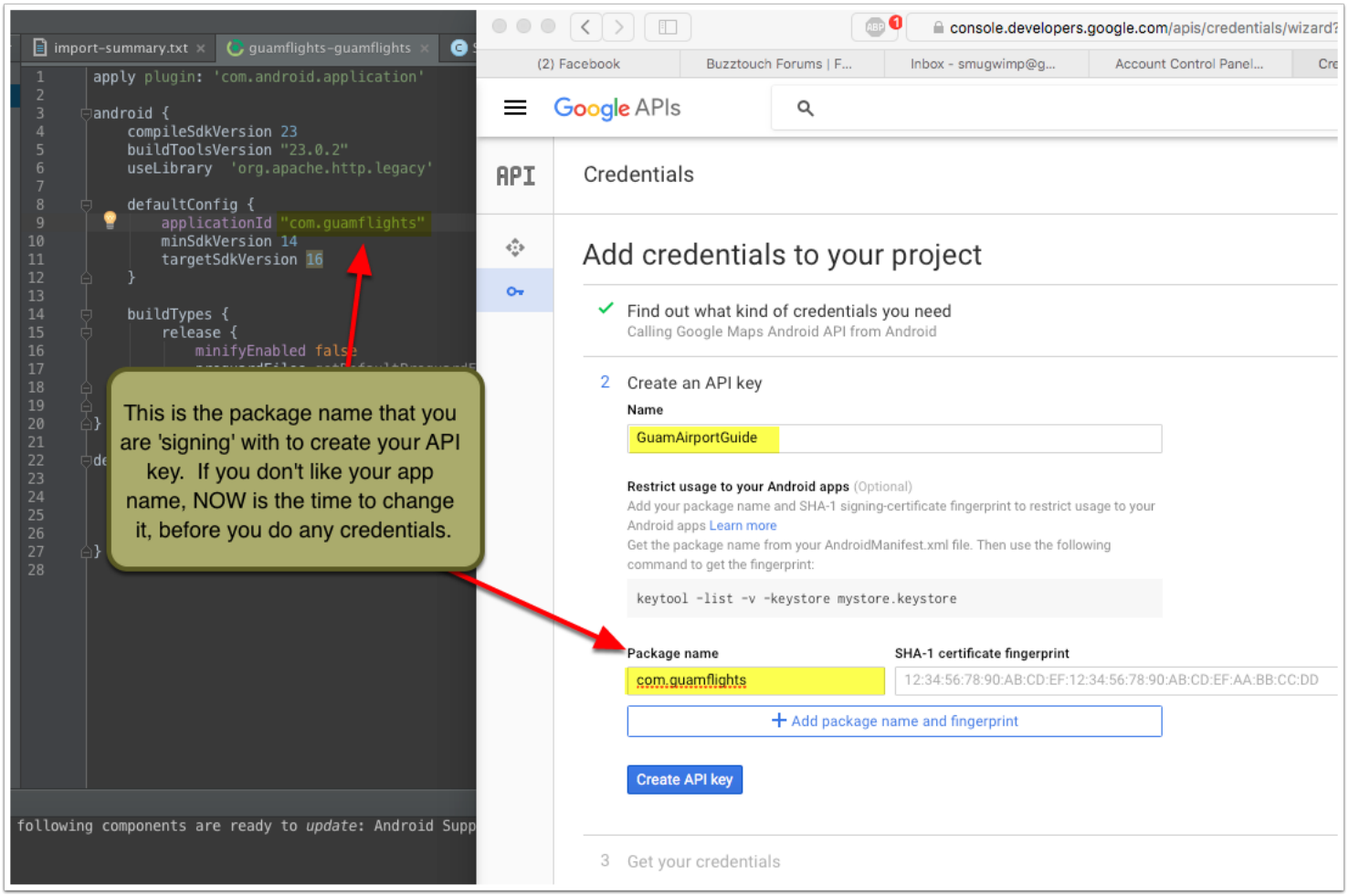
Select the type of API so that correct credentials can be determined

Press "What Credentials Do I Need" when complete.



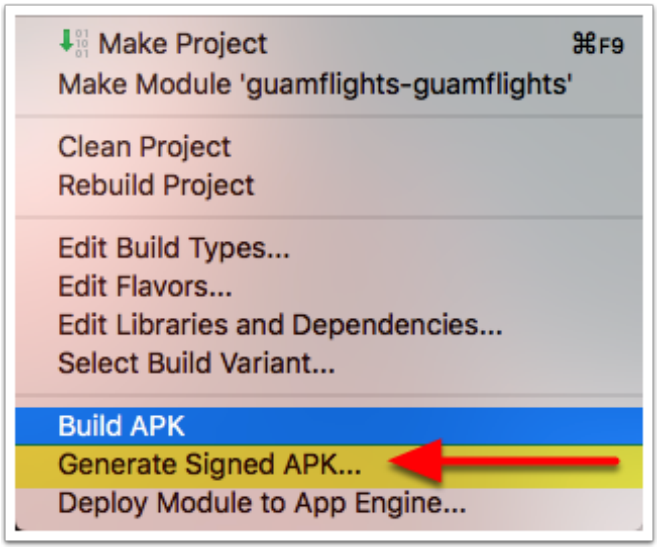
Create Credentials - Ensure correct package name

Before we go too much further, I'd like to mention that if you don't like your package/app name, now is the time to change it. If you create credentials, and then try to change your app name, you'll have to come back, and do all of this over again. It's better to go ahead and change the name first.



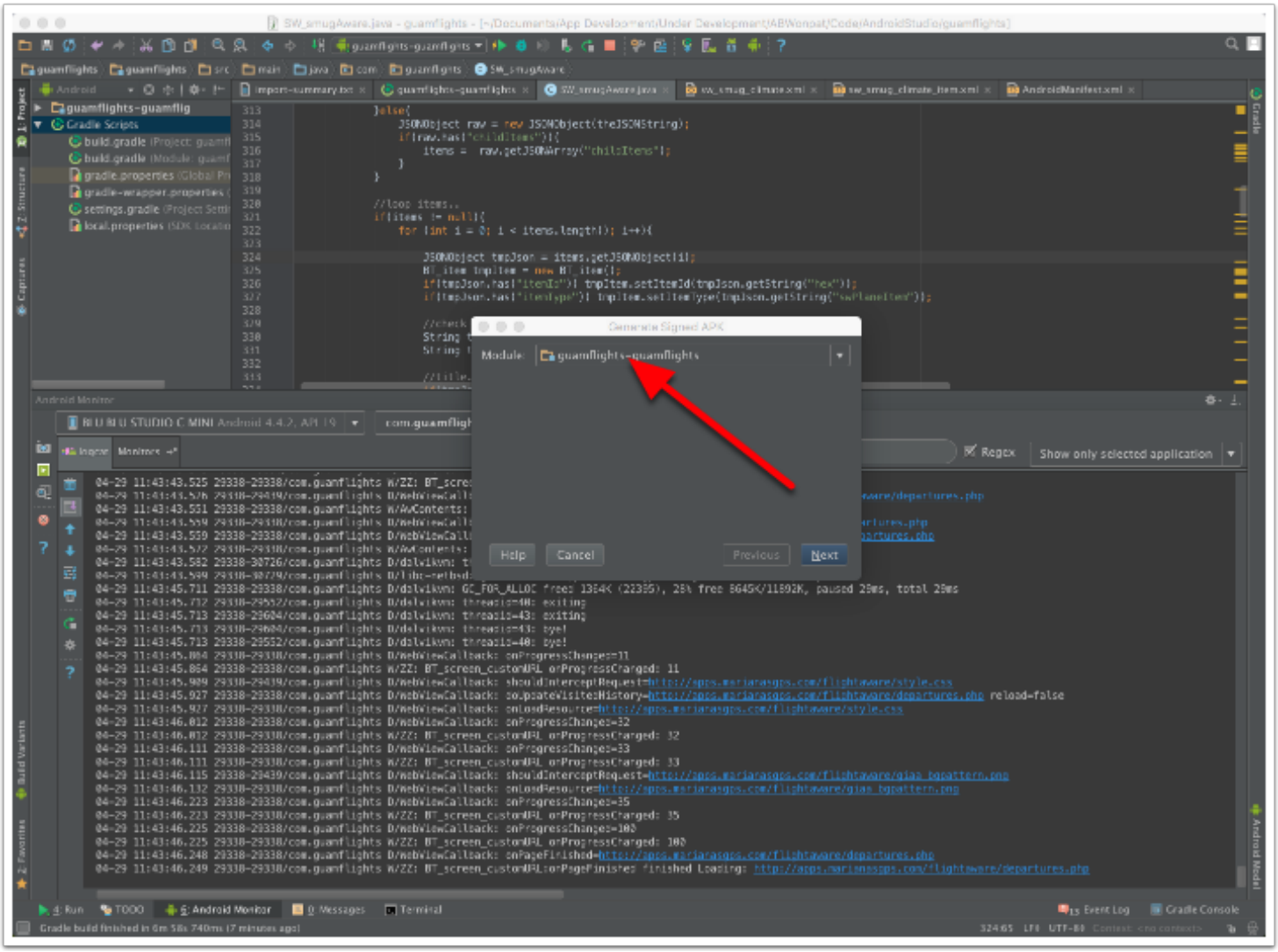
Create a App 'Key' in your release keystore

The easiest way to do this, since I can't seem to find a keytool plugin for Android Studio, Is to create a signed release apk. It doesn't matter if it's your finished product or not; the key won't change, and that's where we need the information from.



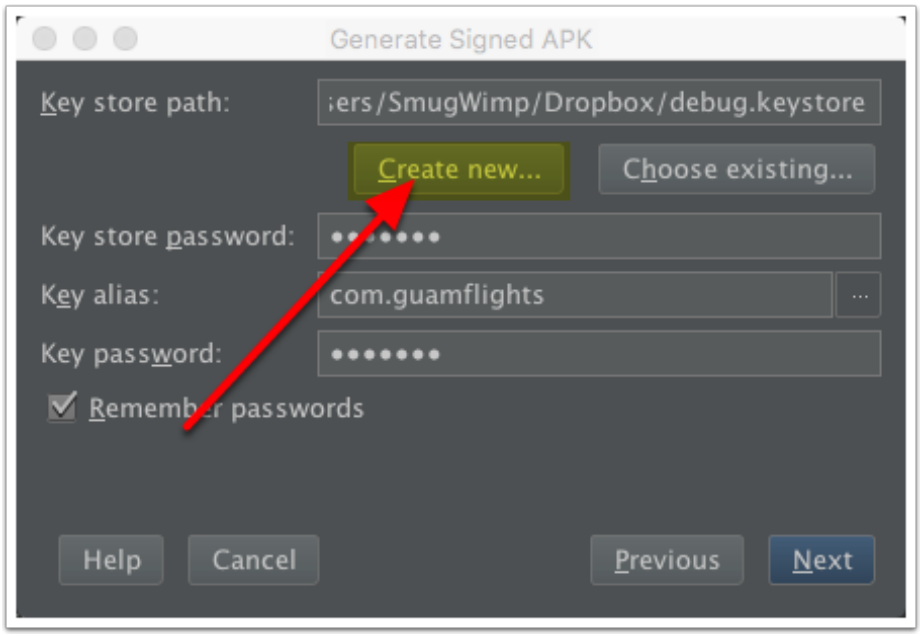
Make sure you're in the right place

It sounds silly, but make sure you're creating the key for the right module. Depending on how complex your app is, sometimes it can come into question.



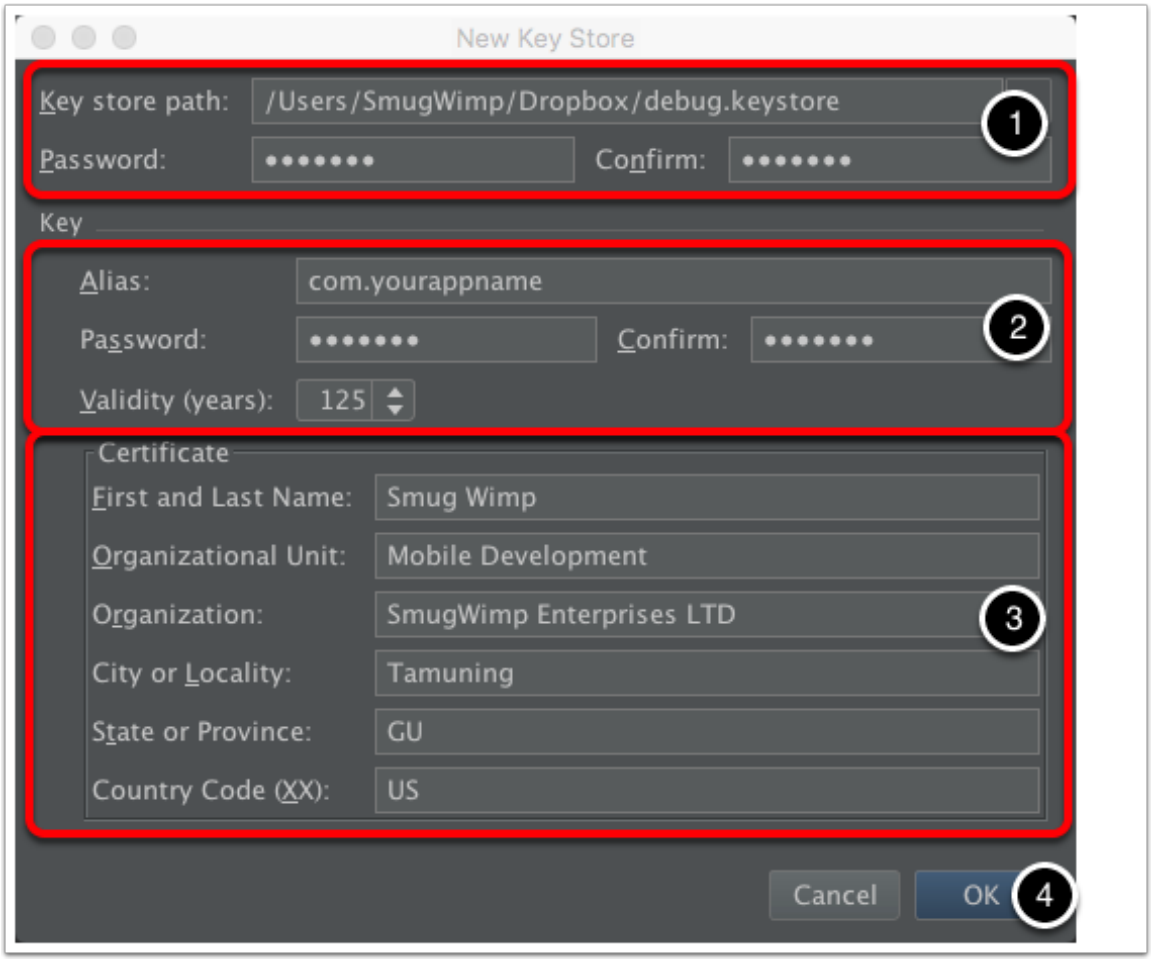
Create New Key

If you haven't created a key for the app, this is where you start. Press the 'Create New' button.



Fill in the information

1. Select the keystore. This is my debug keystore, although I said release earlier. You probably want to navigate and select your release keystore. When you select the proper keystore, enter and confirm the 'keystore' password. This is not the app key password. This is the 'master access' password for the keystore. By default, the debug keystore password is 'android'. "You" create your release keystore, so only you know what it is. And it should stay that way.
2. The details about your app key and alias. The alias can be anything, but keeping it pertinent is always wise. You need to create a password for the app key, and confirm it. Don't ever lose these passwords, or these keystores. Never ever ever. Set the validity to something outrageous.
3. Somewhat personal information for the certificate that needs to be created.
4. Once all that is done, press 'OK'.

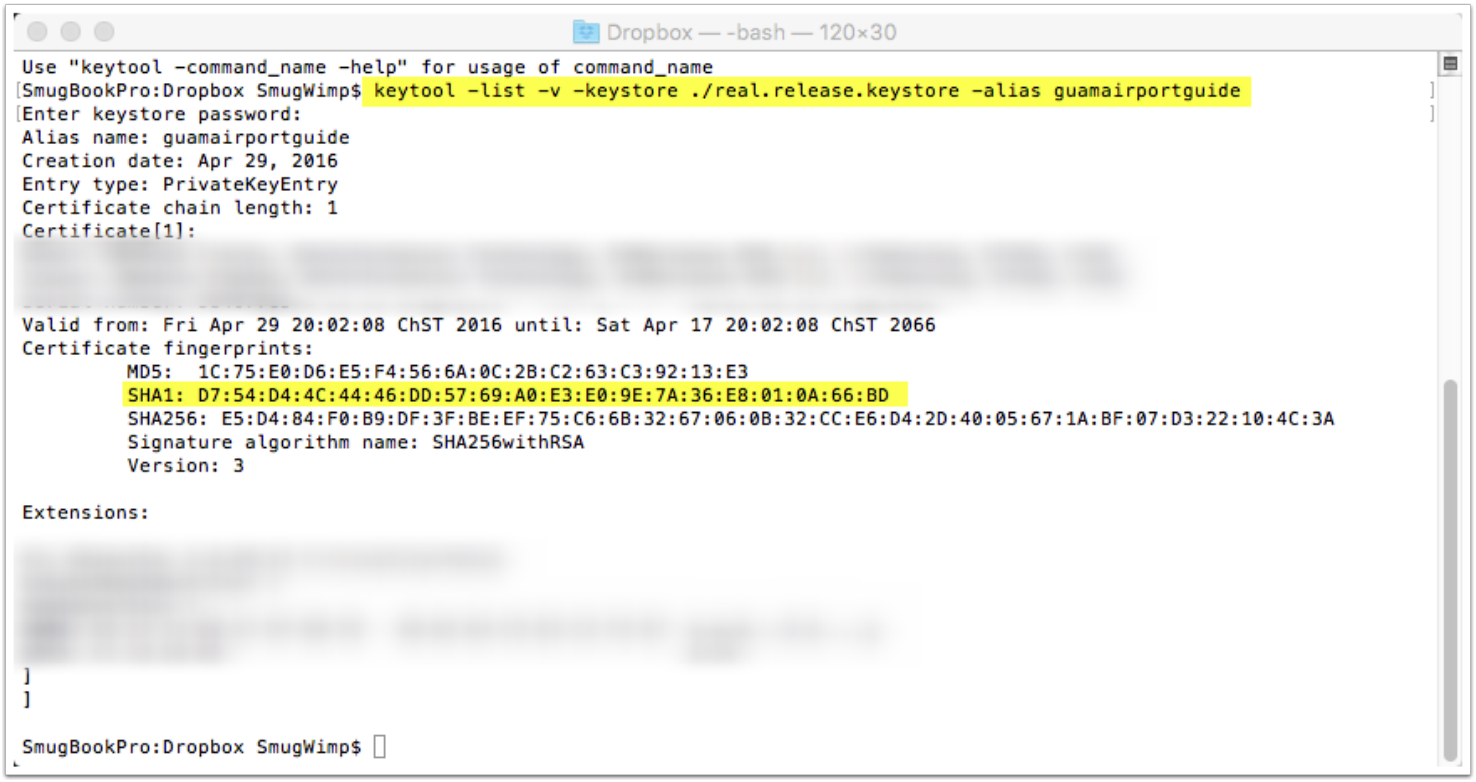


A Quick Note about Keys and Signing...

Open up terminal, and navigate to the same directory as your keystore. From there, type the following command, substituting your correct values:

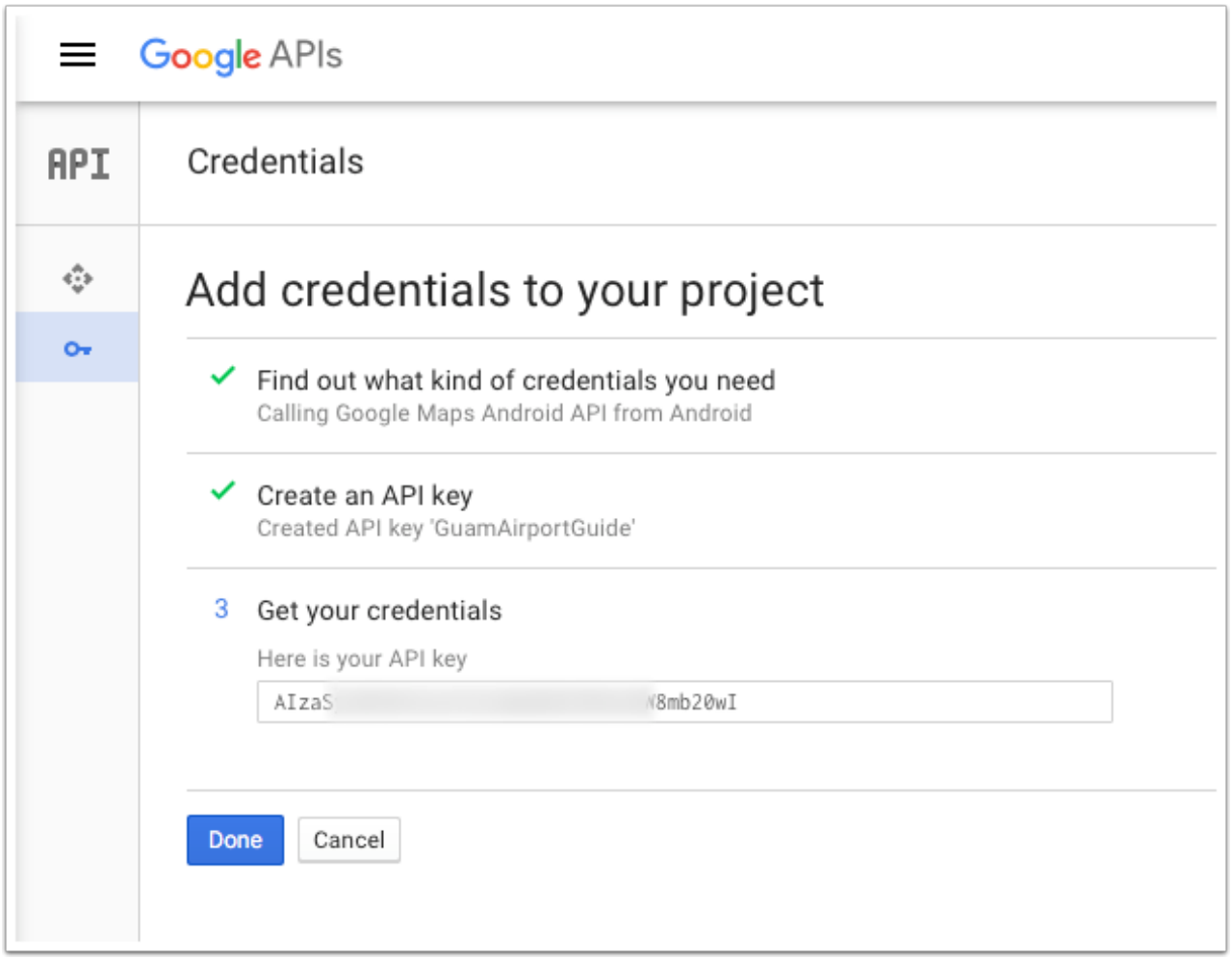
keytool -list -v -keystore your.release.keystore -alias com.yourAppName

Press 'enter' or 'return' or whatever. It should spout out some information about that particular key. Included in that information, is the 'SHA-1 fingerprint' which is what we want.



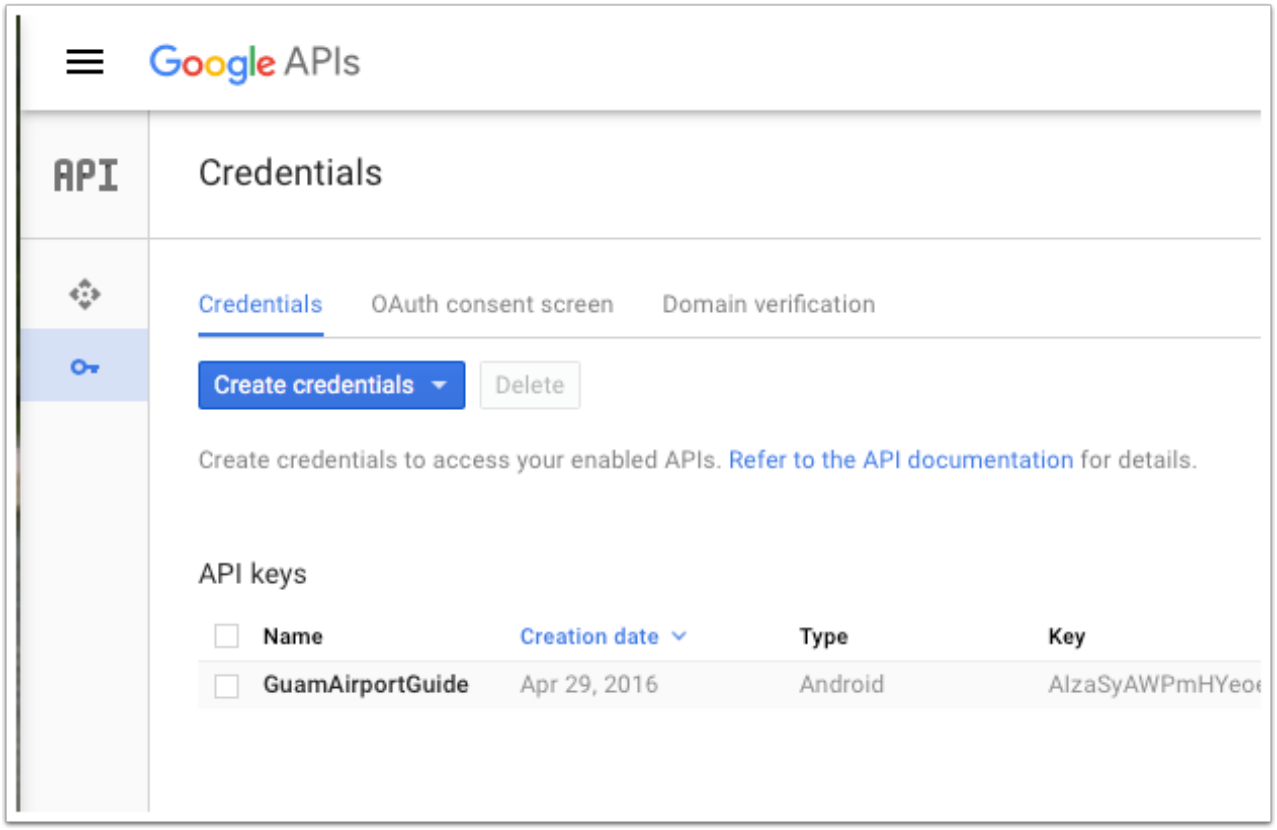
Add your SHA-1 Fingerprint, and press 'create api key'.

Your key will appear in the next section. Copy and paste this into the appropriate places in your app.



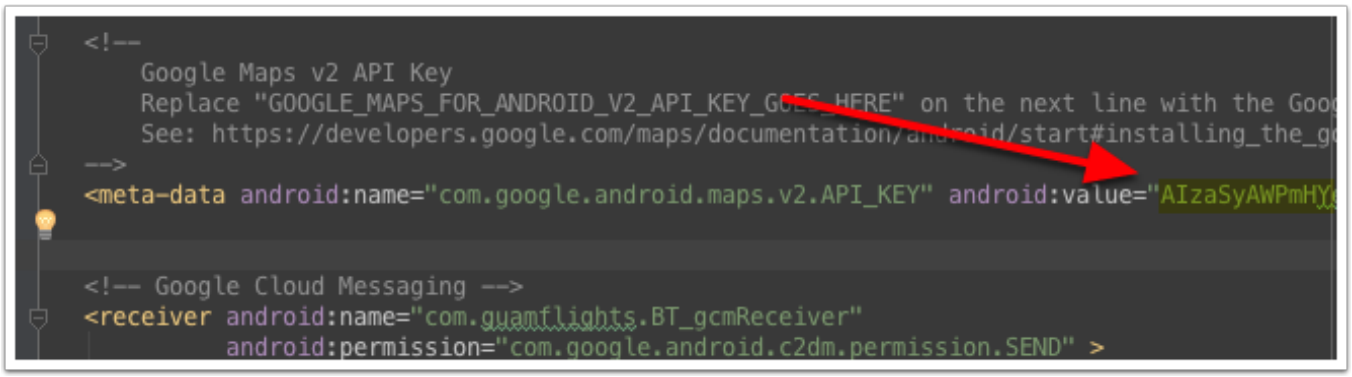
Your API is finished

Now that you're done creating your API key, it's time to put it into your BT App.



1) AndroidManifest

Add the key into your Android Manifest file



Make sure you're referencing the Google Play Services

Make sure that you're referencing the latest Google Play Services in both your Build.Gradle and your AndroidManifest

Add your Google API Key in your Android Manifest

Copy the Google Maps API key provided in your Google Developer Console. Paste in the appropriate location in your AndroidManifest file.

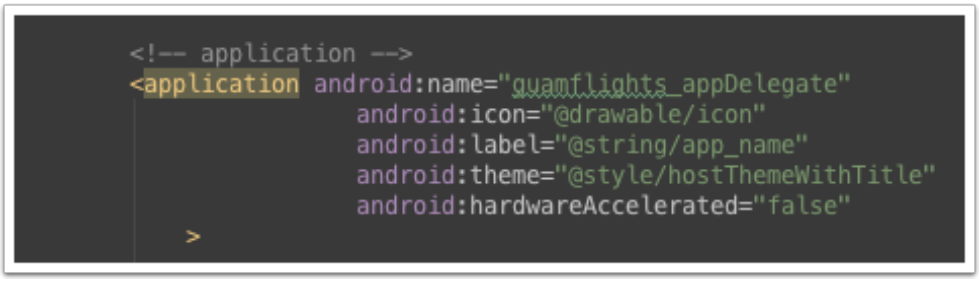
```
<!--
    Google Maps v2 API Key
    Replace "GOOGLE_MAPS_FOR_ANDROID_V2_API_KEY_GOES_HERE" on the next line with the Google Maps for Android API Key provid
    See: https://developers.google.com/maps/documentation/android/start#installing_the_google_maps_android_v2_api
-->
<meta-data android:name="com.google.android.maps.v2.API_KEY" android:value="AIza                                    eM8"/>
```

Remove the debug declaration. It gets annoying.

Remove the line:

android:debuggable="false"

Remember to keep the definition capped ">"



And that is the name of that game.

Provided good omens are upon us, Google Maps should appear in the whatever plugin. If it does not, CHECK THE LOGCAT. That is where the errors will tell you what didn't work, and possibly why. This is important information for troubleshooting, so don't forget.

Good Luck. Happy Appy.

Cheers. If you have questions, don't message me. Post it in the forums, so that all may benefit from your question or problem.