**Getting Started**

This guide is a quick start to adding a map to an Android app. [Android Studio](http://developer.android.com/tools/studio/index.html) is the recommended development environment for building an app with the Google Maps Android API.

**Step 1. Download Android Studio**

Follow the guides to [download](https://developer.android.com/sdk/index.html) and [install](http://developer.android.com/sdk/installing/index.html?pkg=studio) Android Studio.

**Step 2. Install the Google Play services SDK**

Add the [Google Play services package](http://developer.android.com/sdk/installing/adding-packages.html) to Android Studio.

**Step 3. Create a Google Maps project**

Follow these steps to create a new app project including a map activity:

1. Start Android Studio.
2. Create a new project as follows:
   * If you see the **Welcome to Android Studio** dialog, choose **Start a new Android Studio project**, available under 'Quick Start' on the right of the dialog.
   * Otherwise, click **File** in the Android Studio menu bar, then **New**, **New Project**.
3. Enter your app name, company domain, and project location, as prompted. Then click **Next**.
4. Select the form factors you need for your app. If you're not sure what you need, just select **Phone and Tablet**. Then click **Next**.
5. Select **Google Maps Activity** in the 'Add an activity to Mobile' dialog. Then click **Next**.
6. Enter the activity name, layout name and title as prompted. The default values are fine. Then click **Finish**.

Android Studio starts Gradle and builds your project. This may take a few seconds. For more information about creating a project in Android Studio, see the [Android Studio documentation](https://developer.android.com/sdk/installing/create-project.html).

When the build is finished, Android Studio opens the google\_maps\_api.xml and the MapsActivity.java files in the editor. (The file names depend on the configuration values you entered during setup.) Notice that the google\_maps\_api.xml file contains instructions on getting a Google Maps API key before you try to run the application. The next section describes getting the API key in more detail.

**Step 4. Get a Google Maps API key**

Your application needs an API key to access the Google Maps servers. The type of key you need is a **Key for Android applications**. The key is free. You can use it with any of your applications that call the Google Maps Android API, and it supports an unlimited number of users.

Choose **one of the following ways** to get your API key:

* **The fast, easy way:** Use the link provided in the google\_maps\_api.xml file that Android Studio created for you:
  1. Copy the link provided in the google\_maps\_api.xml file and paste it into your browser. The link takes you to the Google Developers Console and supplies information via URL parameters, thus reducing the manual input required from you.
  2. Follow the instructions to create a new project on the console or select an existing project.
  3. Create an Android API key for your console project.
  4. Copy the resulting API key, go back to Android Studio, and paste the API key into the <string> element in the google\_maps\_api.xml file.
* **A slightly less fast way:** Use the credentials provided in the google\_maps\_api.xml file that Android Studio created for you:
  1. Copy the credentials provided in the google\_maps\_api.xml file.
  2. Go to the [Google Developers Console](https://console.developers.google.com/) in your browser.
  3. Use the copied credentials to add your app to an existing API key or to create a new API key. For more details, see the [complete process](https://developers.google.com/maps/documentation/android-api/signup).
* **The full process for getting an API key:** If neither of the above options works for your situation, follow the [complete process](https://developers.google.com/maps/documentation/android-api/signup).

**Note:** If you're using the Google Maps Mobile SDK for Work, you must additionally [enable the Google Maps Mobile SDK for Work](https://developers.google.com/maps/documentation/business/mobile/#the_google_maps_mobile_sdk_for_business_key) for your API project on the Google Developers Console.

**Step 5. Hello Map! Take a look at the code**

Examine the code supplied by the template. In particular, look at the following files in your Android Studio project.

**The XML layout file**

By default, the XML file that defines the app's layout is at res/layout/activity\_maps.xml. It contains the following code:

<fragment xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout\_width="match\_parent"  
    android:layout\_height="match\_parent"  
    android:id="@+id/map"  
    tools:context=".MapsActivity"  
    android:name="com.google.android.gms.maps.SupportMapFragment" />

**The maps activity Java file**

By default, the Java file that defines the maps activity is named MapsActivity.java. It should contain the following code after your package name:

import android.os.Bundle;  
import android.support.v4.app.FragmentActivity;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
import com.google.android.gms.maps.model.LatLng;  
import com.google.android.gms.maps.model.MarkerOptions;  
  
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity\_maps);  
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()  
                .findFragmentById(R.id.map);  
        mapFragment.getMapAsync(this);  
    }  
  
    @Override  
    public void onMapReady(GoogleMap map) {  
        // Add a marker in Sydney, Australia, and move the camera.  
        LatLng sydney = new LatLng(-34, 151);  
        map.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));  
        map.moveCamera(CameraUpdateFactory.newLatLng(sydney));  
    }  
}

If your maps activity file doesn't contain the above code, replace the contents of the file with the above code, after your package name.

**Step 6. Connect an Android device**

The simplest way to see your app in action is to connect an Android device to your computer. Follow the [instructions](http://developer.android.com/tools/device.html) to enable developer options on your Android device and configure your application and system to detect the device.

Alternatively, you can use the Android Emulator to run your app. Use the [Android Virtual Device (AVD) Manager](http://developer.android.com/tools/devices/managing-avds.html) to configure one or more virtual devices which you'll be able to use with the Android Emulator when you build and run your app. You need an emulator image with the Google APIs platform based on Android 4.2.2 or higher. Take note of the instructions for [configuring virtual machine acceleration](http://developer.android.com/tools/devices/emulator.html#accel-vm), which you should use with an **x86 target AVD** as described in the instructions. This will improve your experience with the emulator.

**Step 7. Build and run your app**

In Android Studio, click the **Run** menu option (or the play button icon) to run your app.

When prompted to choose a device, choose one of the following options:

* Select the Android device that's connected to your computer.
* Alternatively, select the **Launch emulator** radio button and choose the virtual device that you've previously configured.

Click **OK**. Android Studio will invoke Gradle to build your app, and then display the results on the device or on the emulator. It could take a couple of minutes before the app opens.