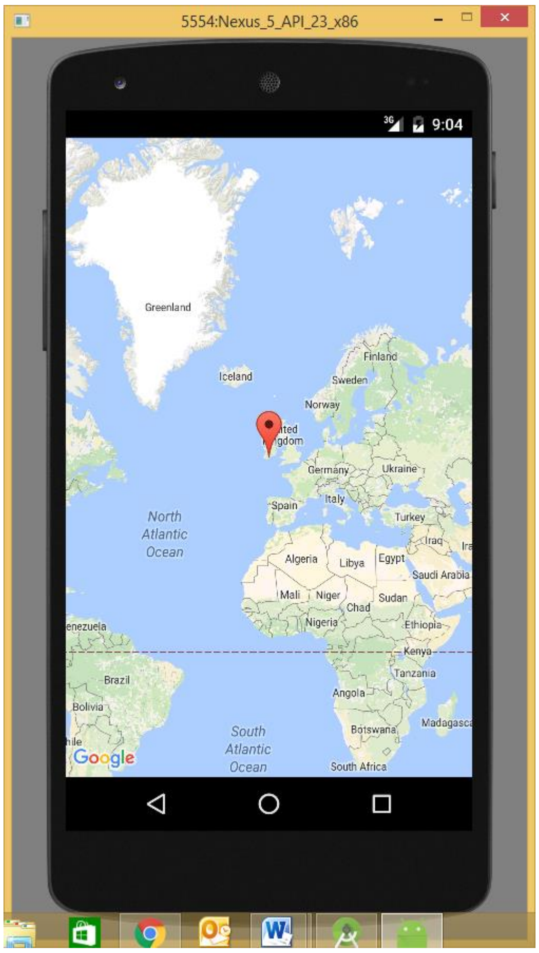
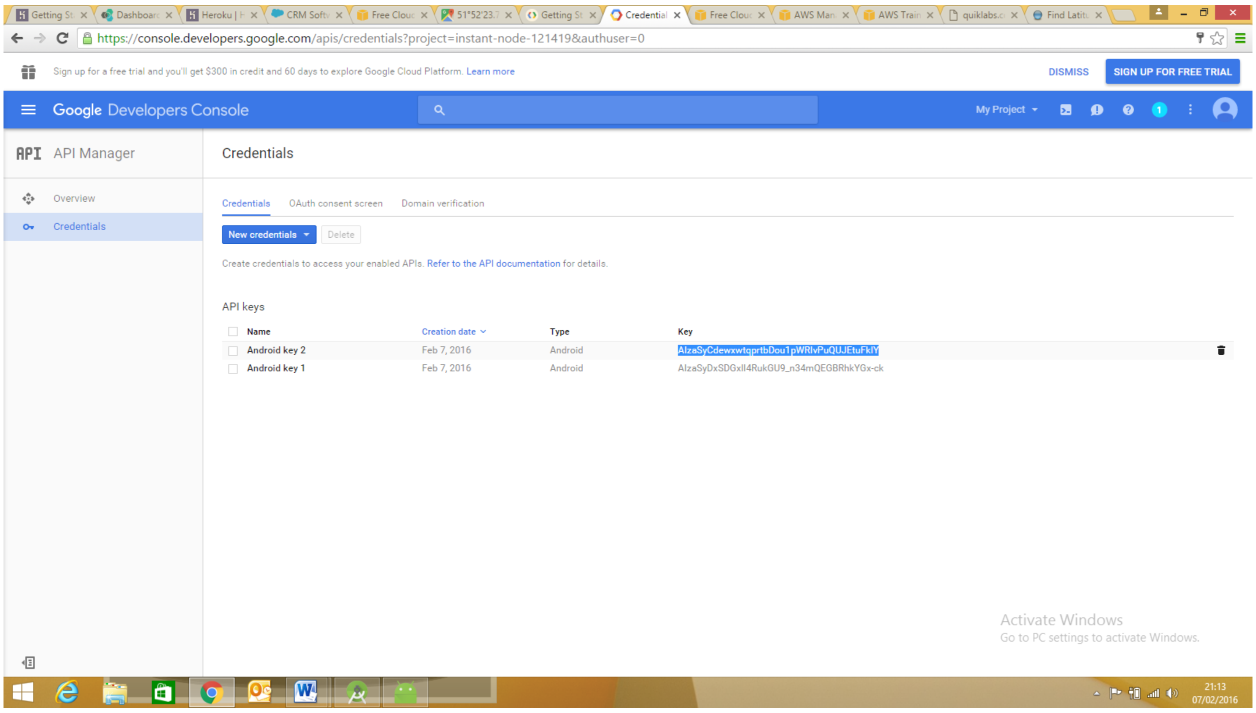
Google Maps API



- Create a gmail account if you don't have one already
- Create a Google Maps activity in Android Studio
- You can choose the exact location using http://www.findlatitudeandlongitude.com/
- Log into https://console.developers.google.com/apis/credentials?project=instant-node-121419&authuser=0 to get an API key which connects your app to Google services as shown below.



https://console.developers.google.com/apis/credentials?project=instant-node-121419&authuser=0

Sign up for a free trial and you'll get $300 in credit and 60 days to explore Google Cloud Platform. Learn more

Credentials

Create credentials to access your enabled APIs. Refer to the API documentation for details.

API keys

Name	Creation date	Type	Key
Android key 2	Feb 7, 2016	Android	AIzaSyCdewxwtqprtbDou1pWRIvPuQUJEtuFkIY
Android key 1	Feb 7, 2016	Android	AIzaSyDxSDGxlI4RukGU9_n34mQEGBRhkYGx-ck

Getting Started

https://developers.google.com/maps/documentation/android-api/start

This guide is a quick start to adding a map to an Android app. Android Studio 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 and install Android Studio.

Step 2. Install the Google Play services SDK

Add the Google Play services package 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.

When the build is finished, Android Studio opens the `google_maps_api.xml` and the `MapsActivity.java` files in the editor. (Note that your activity may have a different name, but it will be the one you configured 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 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.

- **The full process for getting an API key:** If neither of the above options works for your situation, follow thecomplete process.

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 {

    private GoogleMap mMap;

    @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 googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney, Australia, and move the camera.
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in
Sydney"));
        mMap.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 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 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. When choosing your emulator, ensure that you use Android 4.2.2 or higher, and be careful to pick an image that includes the Google APIs, or the application will not have the requisite runtime APIs in order to execute. Also, take note of the instructions for configuring virtual machine acceleration, 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.

You should see a map with a marker positioned over Sydney, Australia. If you don't see a map, confirm that you've completed all the steps described on this page. In particular, check that you've added an API key as described above.

Next steps

You may wish to look at some sample code.

You can read more about map objects in the developer's guide.