

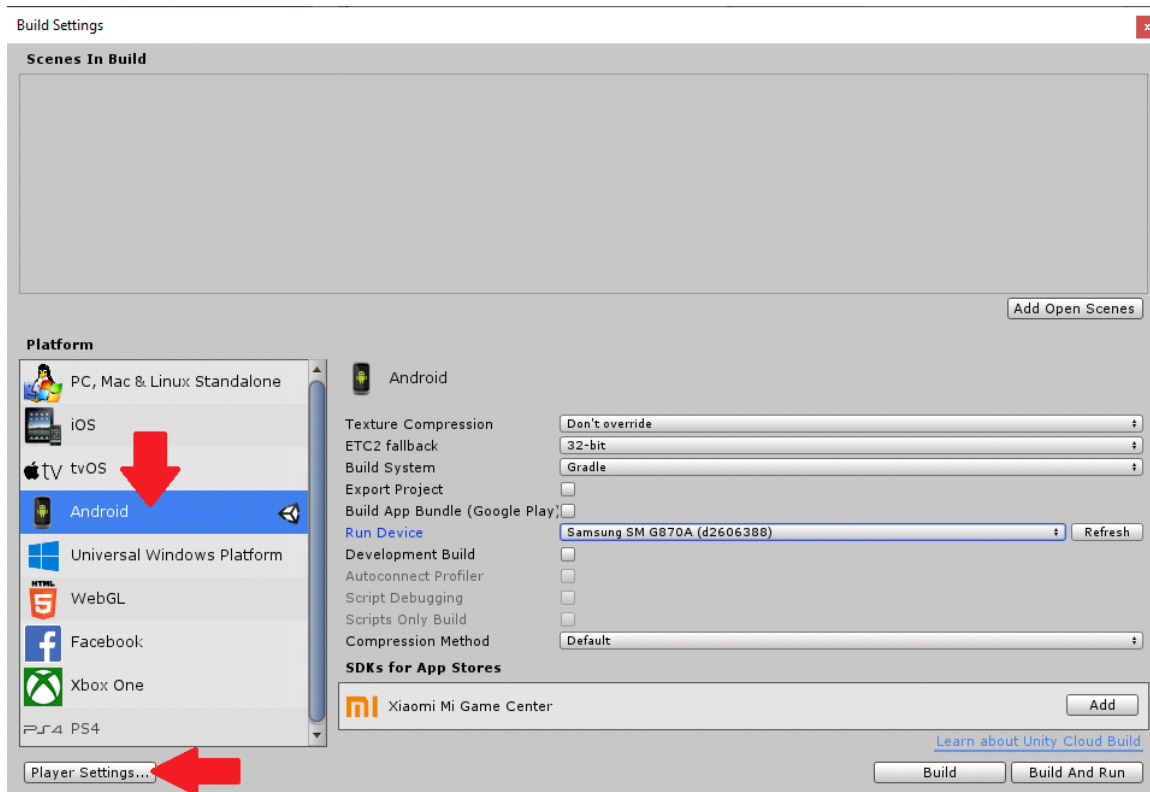
# Google Authentication with Braincloud

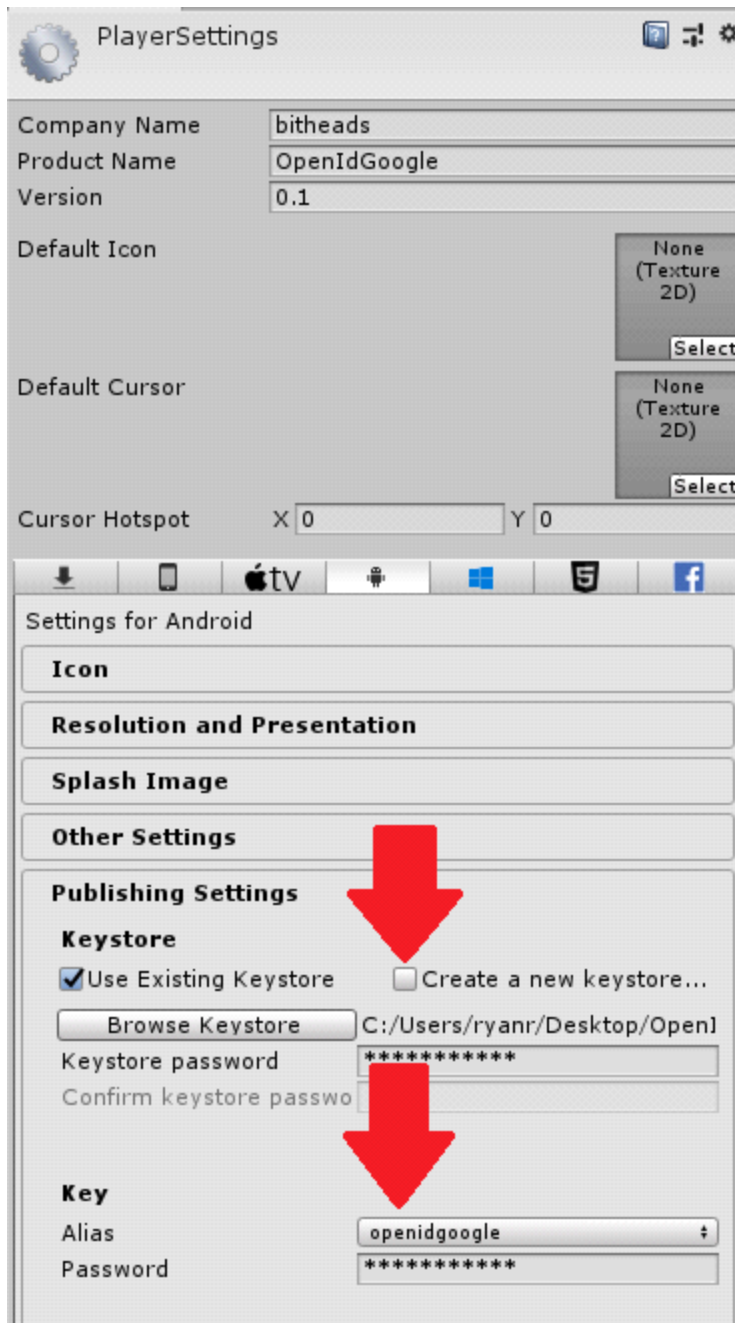
## The old method, using Google Play Games

This tutorial is for Android, because Google Play Games only supports Android.

### Setup an android app in Unity, and prepare the Package name and SHA-1

1. create unity project, go to File > Build Settings, and change the platform to Android. Then on the bottom left of Build Settings, go into Player Settings and under Publishing Settings, make a new signing keystore and get the SHA-1. Make sure to also add an alias to your key.





To get the SHA-1 follow google's tutorial here:

<https://support.google.com/cloud/answer/6158849?hl=en#installedapplications&android>

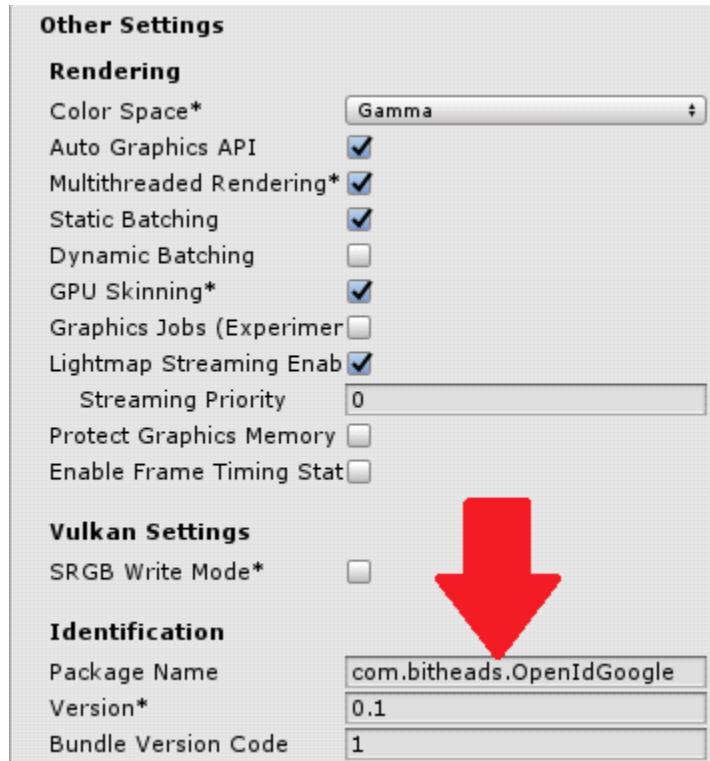
***Tips for SHA-1 tutorial :***

***- Start at step 2***

***- If you're having trouble finding the keytool.exe, try looking in your jdk folder.***

**- If the command in step 2 isn't working, make sure you're in the right directory of the keytool in the terminal before running the command.**

You will also need to make a package name for your project which can also be done in the Player Settings. Make sure it matches the Company name you chose and Product name in the Player Settings.



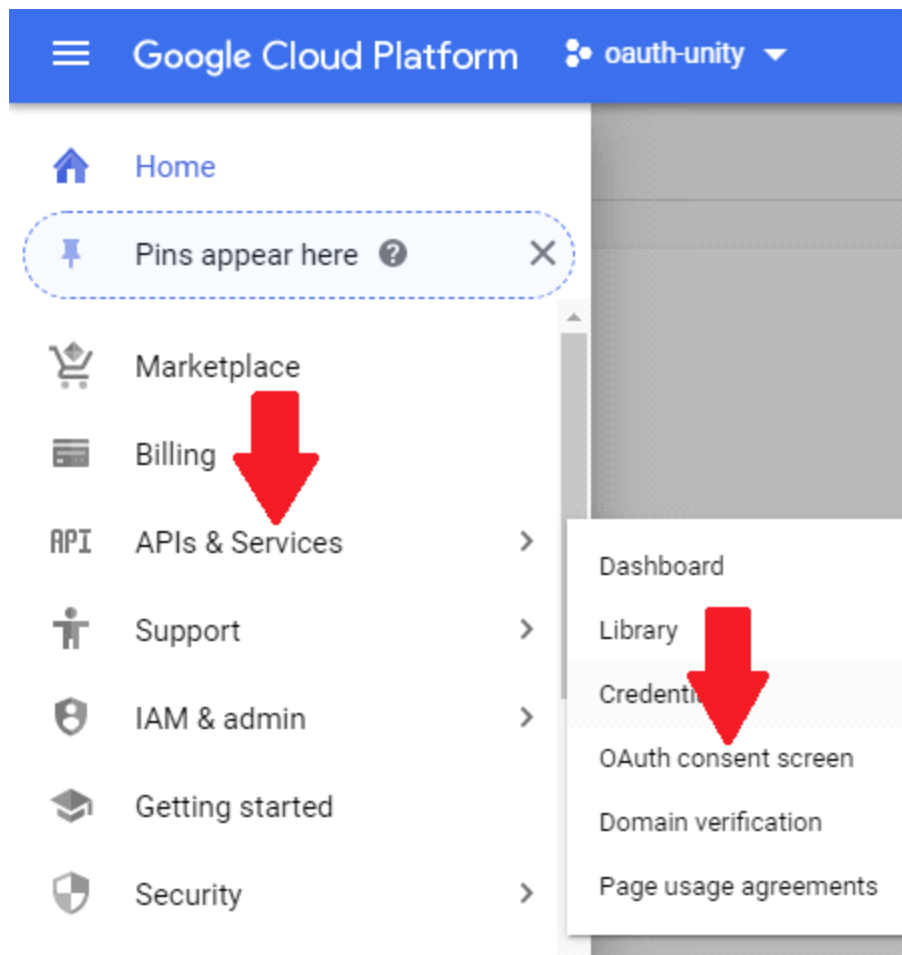
Save your SHA-1 and package name for later, you will need it.

## Setup your app on google cloud platform

1. go to google cloud platform <https://console.cloud.google.com/> and create a new app. Name it whatever you want.



2. Now go into your newly created project and hit the APIs & Services, and go to OAuth consent screen




3. In the OAuth Consent Screen, you will likely need to make an external app, since internal apps require you to be a G Suite User.

## OAuth consent screen


---

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

### User Type

☐ Internal 

Only available to users within your organization. You will not need to submit your app for verification.

☒ External 

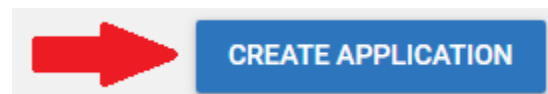
Available to any user with a Google Account.



4. On the next page, put in an application name, and a support e-mail. Then scroll to the bottom and press save. DO NOT upload an application logo as you will require a tonne of more requirements to get your app out of a state of needing verification permanently.

## Setup your app on Google Play Services

1. Open up Google Play Console <https://play.google.com/apps/publish/> and create an application. Choose a title and click create.




### Create application

Default language \*


English (United States) – en-US ▼

Title \*
























0/50

CANCEL

CREATE

2. Go into your app, and in the drop down click on Services and APIs

← All applications

-  Dashboard
-  App releases 
-  Android Instant Apps
-  Artifact library
-  Device catalog
-  App signing
-  Store listing 
-  Custom store listings
-  Content rating 
-  App content 
-  Pricing & distribution 
-  In-app products
-  Translation service
-  **Services & APIs** 
-  Optimization tips

3. You will want to enable google play services for this app, so click on the button to USE GOOGLE PLAY GAMES SERVICES IN THIS APP

#### Google Play game services

Google Play game services let you add gaming features to your games on Android. Users can view leaderboards to compare their scores with others, unlock achievements, and sync their game progress across their devices. [Learn more](#)

USE GOOGLE PLAY GAME SERVICES IN THIS APP

Want to know a bit more before you start?

[See a step-by-step guide for using Google Play games services](#)

4. It will bring you to the game services screen where you will want to ADD NEW GAME, then give it a name and category.



Check the header "I already use Google API's in my game" and see if you can find your app in the drop down options. It should be there since we created a project for the game on the Google Developers Console in previous steps.



## Set up Google Play game services for an app

Do you already use Google APIs in your app?


[I don't use any Google APIs in my game yet](#)

[I already use Google APIs in my game](#)

If your game already uses Google APIs, you have created a project for the game on the [Google Developers Console](#) before.

Choose an API console project to link

GoogleTutorial
oauth-unity



[I can't see my project in the list](#)

What kind of game is it?

Choose a category ▼

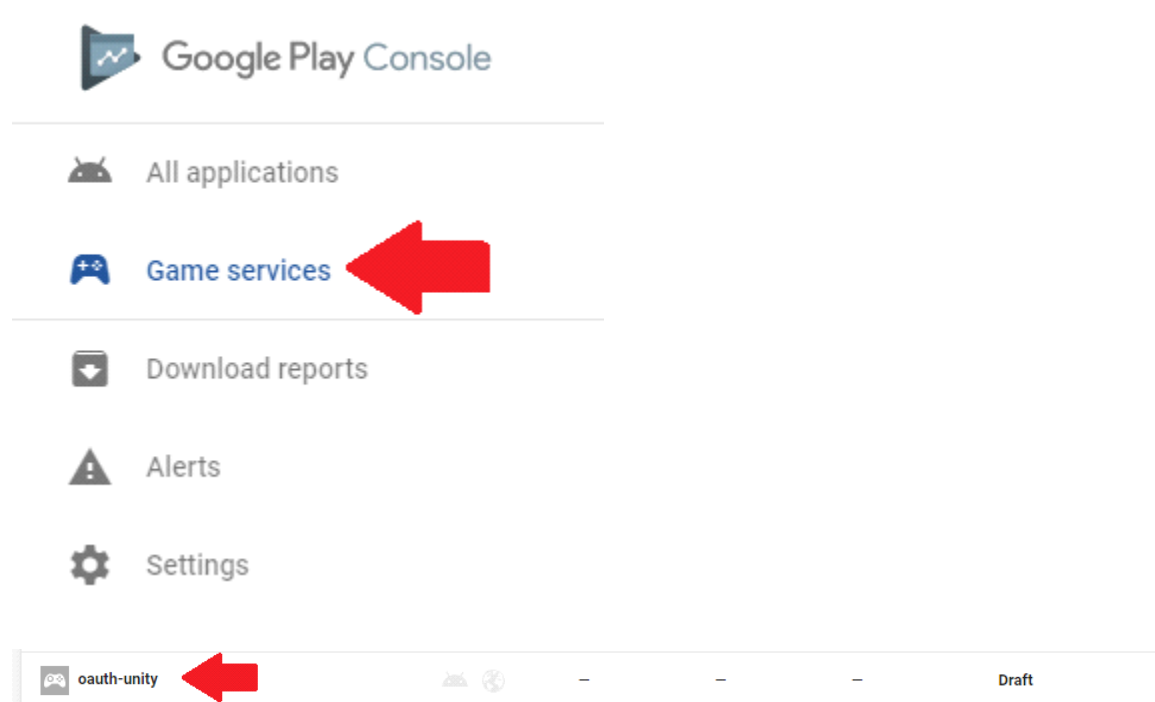


The category helps users browse interesting games.

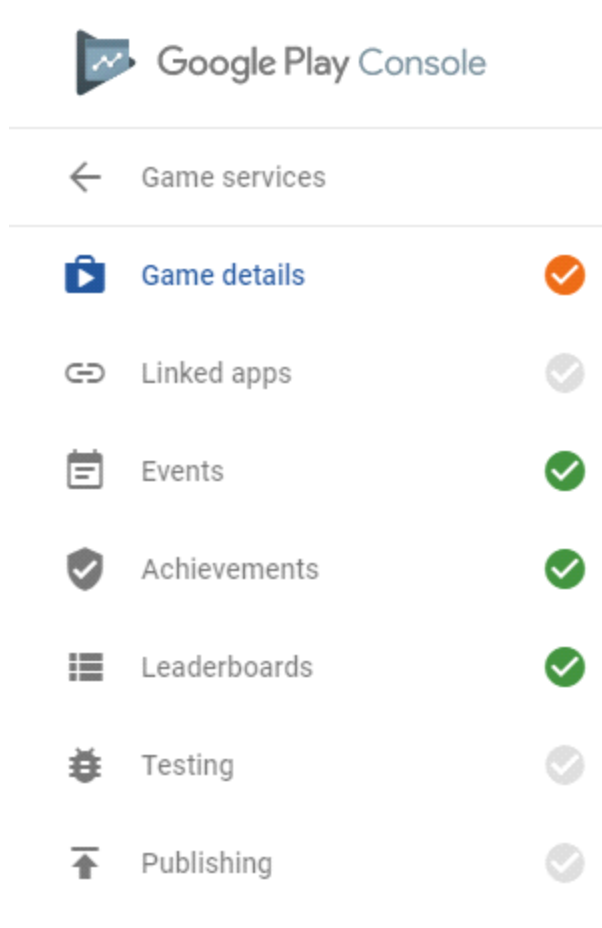
CONTINUE



5. Now under the Game Services Tab, click on your newly created game



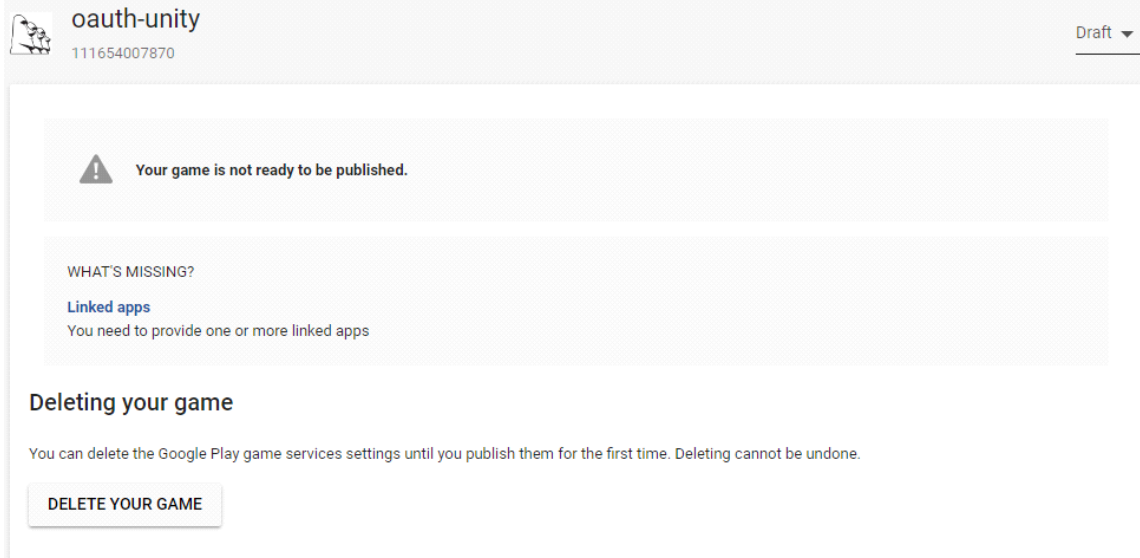
6. Now you'll notice that you have some things to do before your app is ready. You'll need to go through each of the checkmarks until they are green. The events, achievements and leaderboards are features you can add, but don't need to be added, so they are green by default.



7. In Game details, simply fill out your information about your app.

8. We will get back to linked apps after we configure a Google Sign In web app.

9. In Testing, be sure to add some familiar e-mails that you will use to login to google with, for testing purposes. You'll notice once you move to the next step in testing, it will ask you to make at least one linked app.



## Configure a Google Sign In App

Go to the Google Sign In plugin page on github and download the latest release of the plugin.

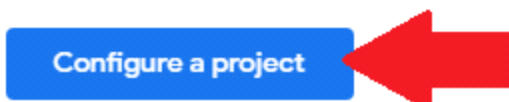
<https://github.com/googlesamples/google-signin-unity>

You will also need to do **STEP 2** of Configure a Google API project here:

<https://developers.google.com/identity/sign-in/android/start> which is mentioned under "Configuring the application on the API Console". Step 2 of this tutorial is the most important as it will generate the web client we need on the google cloud platform. During the process it will ask what type of app is calling it, choose ANDROID and put in the info of your app, this will generate both android and a web client credential. You only have to do STEP 2.

## 2 Configure a Google API project

To use the sample, you need to provide some additional information to finish setting up your project. Click the button below, and specify the package name `com.google.samples.quickstart.signin` when prompted. You will also need to provide the SHA-1 hash of your signing certificate. See [Authenticating Your Client](#) for information.



Go back to your Google Cloud Platform project, and go under APIs and Services, and Credentials. You will notice there is a WebApp that has been auto generated by Google.

Credentials

Create credentials ▾ Delete

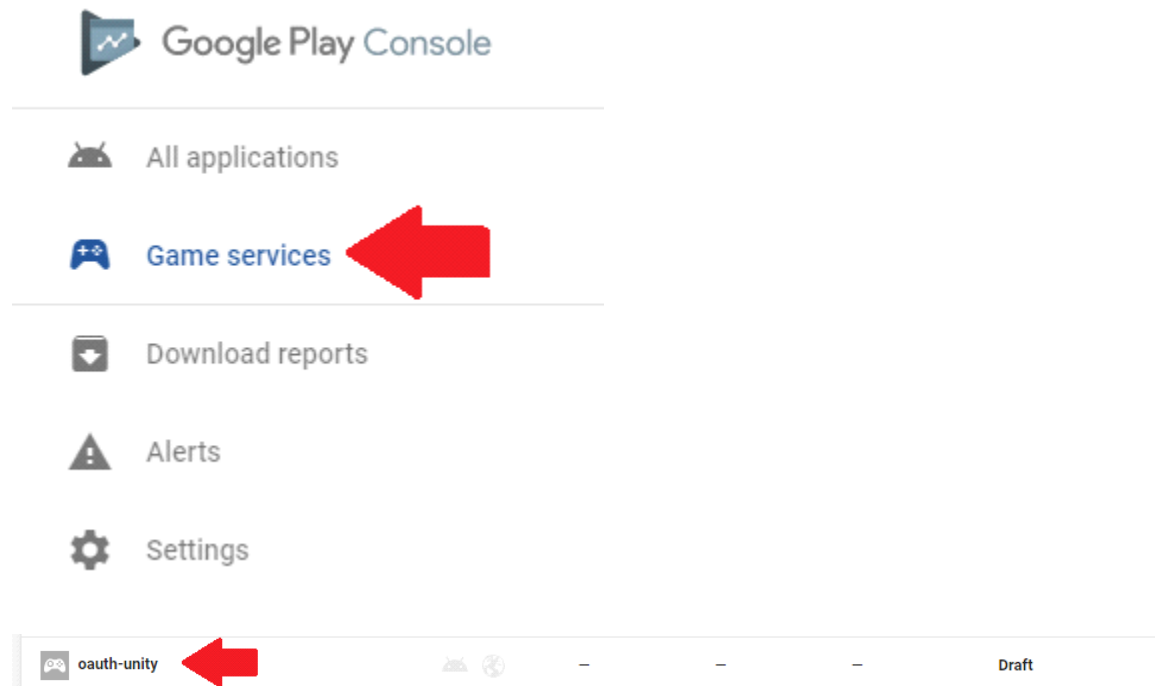
Create credentials to access your enabled APIs. For more information, see the [authentication documentation](#).

OAuth 2.0 client IDs

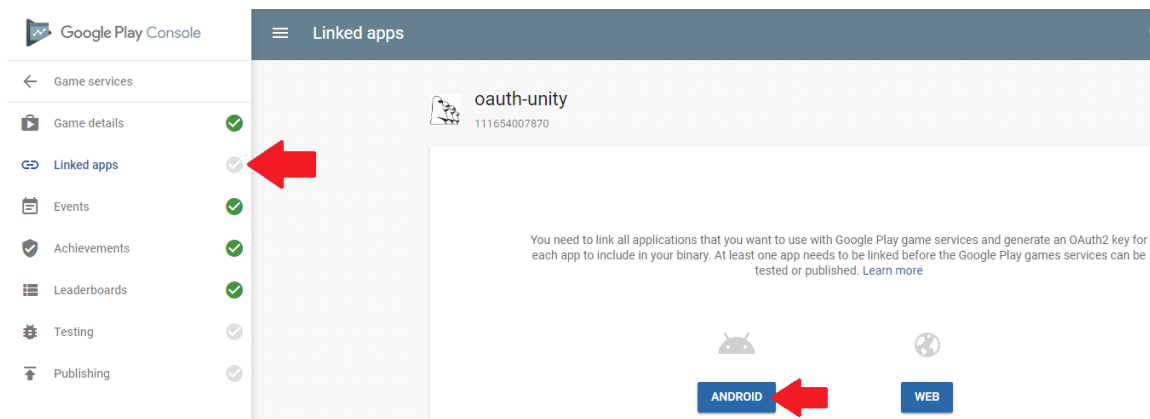
<input type="checkbox"/> Name	Creation date ▾	Type	Client ID
<input type="checkbox"/> OAuth client	Jan 10, 2020	Android	780423637529-dd18o27qo6pafj0l217uak8hpsiq9ntp.apps.googleusercontent.com
<input type="checkbox"/> Web client (Auto-created for Google Sign-in)	Jan 10, 2020	Web application	780423637529-hcg9gdb04egbbh7h6k9pkukd8a9j9f8u.apps.googleusercontent.com

## Link the app on Google play


1. Go back to Google play <https://play.google.com/>, and under the Game Services tab, choose your app once again.



2. Go under Linked Apps Tab, and choose Android.



On the next page, make sure the name of the app stays consistent.

[<](#)  **Link an Android app** Save and continue

Step 1: Enter the app details


ENGLISH (UNITED STATES) – en-US

Name of the app

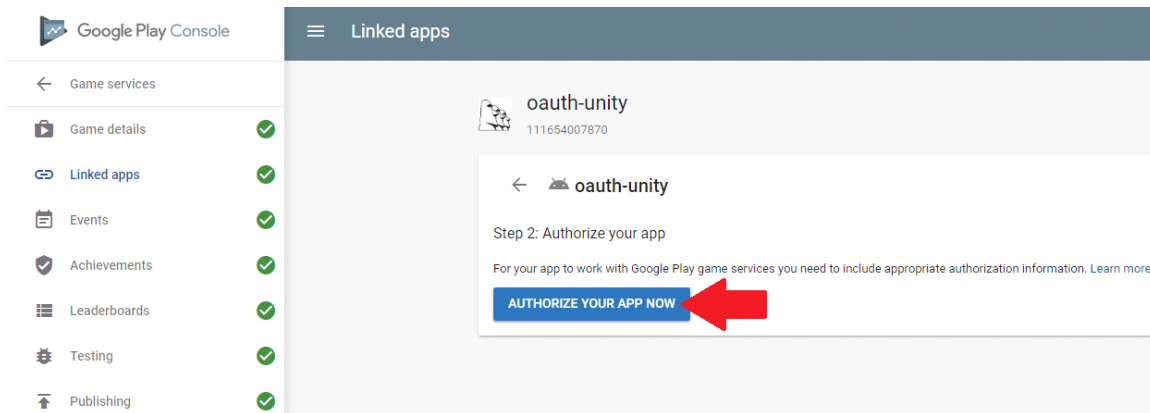
English (United States) – en-US

oauth-unity




Package name

 com.bitheads.OauthUnity

Fill in the appropriate title and package name of your project. Once you hit save and continue, things should be looking ready.



It may ask you to authorize your app in which case just use the information you've been gathering like the SHA-1 and put it in the fields. Otherwise, you're good!

Name	Details	OAuth 2.0 Client	Status
 <b>oauth-unity</b>	com.bitheads.OauthUnity		 Ready to publish

NOTE: the appId of the linked app MUST and WILL MATCH the app id of the android app that was generated on your google cloud console.

## Adjust Settings in Braincloud

When you create your app on Braincloud, make sure to include Google as one of the platforms. If you have already made your App, you can turn on Google platform support by going to Core App Info> Platforms and selecting Google Android from the list.

## New App



### App Name

Enter a display name for your app.


### Enable Game Features?

Enable features specific to games and gamified apps, such as achievements, leaderboards and multiplayer. Settings can be changed in **Core App Info**.

- ☒ Yes, of course!  
☐ No.

### Supported Platforms

Select at least one platform supported by your app.

- |   |  |  |
|---|--|--|
| <input checked="" type="checkbox"/> Apple iOS | <input type="checkbox"/> Apple tvOS          | <input type="checkbox"/> Apple watchOS   |
| <input type="checkbox"/> BlackBerry           | <input checked="" type="checkbox"/> Facebook |  <input checked="" type="checkbox"/> Google Android |
| <input type="checkbox"/> Linux                | <input type="checkbox"/> Mac OS X            | <input type="checkbox"/> PlayStation 3   |
| <input type="checkbox"/> PlayStation 4        | <input type="checkbox"/> PlayStation Vita    | <input type="checkbox"/> Roku streaming player   |
| <input type="checkbox"/> Tizen                | <input type="checkbox"/> Unknown             | <input type="checkbox"/> Web   |
| <input type="checkbox"/> Wii                  | <input checked="" type="checkbox"/> Windows  | <input type="checkbox"/> Windows Phone   |
| <input type="checkbox"/> Xbox 360             | <input type="checkbox"/> Xbox One            |  |

✕ Cancel

✓ Save

Next, go to Design tab, and go to Configure Platforms.

The Google package name is the package name of your project.



## Configure Platforms

Apple

Facebook

Google

Steam

Twitter

Google Service Account Email

Google Package Name

com.bitheads.OauthUnity

Google App ID

Google Client ID

Google Client Secret

Google Service Account p12 Certificate

Select a certificate file

✗ Not Configured

The Google AppId and Client Id is the ID of the Web App that was generated by Google Sign in when you were configuring your project for it. To see this go back to <https://console.cloud.google.com/>, go to your app, Under the API & Services tab, go to Credentials, click on your Web App, and take the Client ID. Put it into the Client ID section in braincloud... Then take the number in front and put that as the Google App ID in braincloud.

## Credentials

Create credentials ▼

Create credentials to access your enabled APIs. For more information, see the [authentication documentation](#).

## OAuth 2.0 client IDs

<input type="checkbox"/>	Name	Creation date	Type	Client ID
<input type="checkbox"/>	OAuth client	Jan 16, 2020	Android	10...
<input type="checkbox"/>	Web client (Auto-created for Google Sign-in)	Jan 16, 2020	Web application	10...

- The Google Secret is the Web App's secret.

Client ID	223329023619-7o4r8l0qe0ijnntpv1fgghkf98ve76ch.apps.googleusercontent.com
Client secret	[REDACTED]
Creation date	Dec 18, 2019, 11:10:18 AM

## Last steps

1. Add The latest of

braincloud plugin : <https://github.com/getbraincloud/braincloud-csharp>

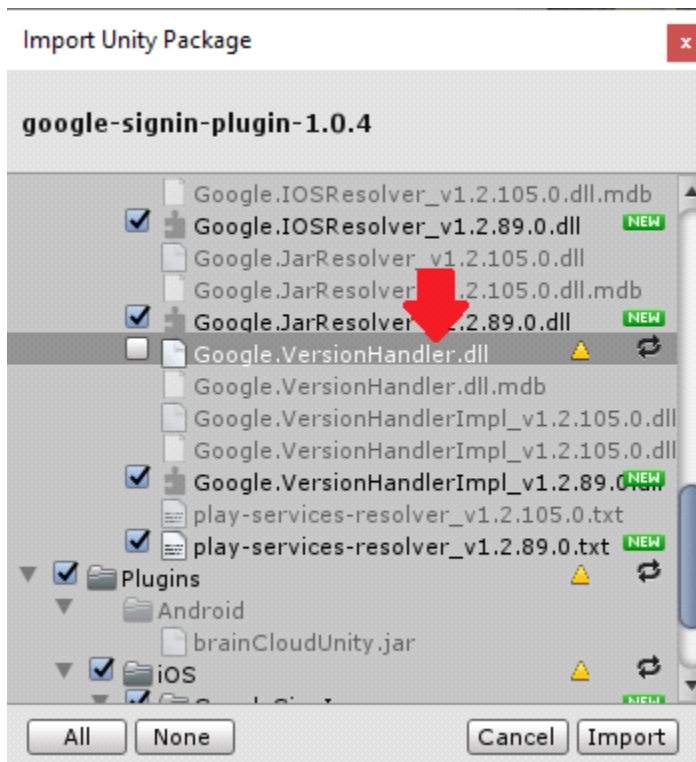
google sign in unity plugin: <https://github.com/googlesamples/google-signin-unity>

and version v0.9.64 google play plugin :

<https://github.com/playgameservices/play-games-plugin-for-unity> (since the latest releases cause major bugs at the moment)

into your unity project.

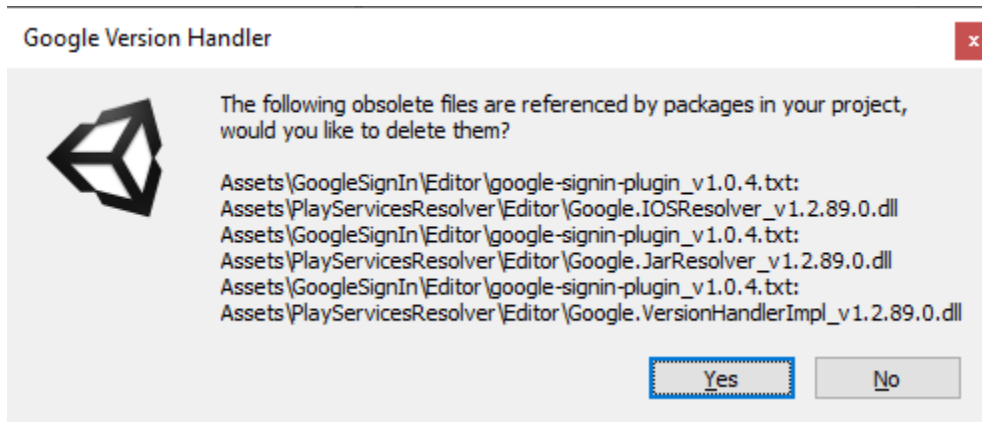
NOTE: google play plugin and google sign in unity plugin crossover in files. Be sure to turn this check off when importing the google sign in unity plugin into your project to avoid conflicting .dlls.



You will then notice that Unity will start throwing errors. They are there because there are some conflicting defines in the Parse folder that the Google Sign In plugin introduces. The fix is to delete the newly created Parse folder.

```
[14:12:24] Assets\GoogleSignIn\Future.cs(72,40): error CS0433: The type 'TaskCompletionSource<T>' exists in both 'Unity.Tasks, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' and 'netstandard, Version=2.0.0.0, Culture=neutral, PublicKeyToken=cc7b13ffcd2dd51'
[14:12:24] Assets\GoogleSignIn\GoogleSignIn.cs(117,12): error CS0433: The type 'Task<T>' exists in both 'Unity.Tasks, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' and 'netstandard, Version=2.0.0.0, Culture=neutral, PublicKeyToken=cc7b13ffcd2dd51'
[14:12:24] Assets\GoogleSignIn\GoogleSignIn.cs(130,12): error CS0433: The type 'Task<T>' exists in both 'Unity.Tasks, Version=0.0.0.0, Culture=neutral, PublicKeyToken=null' and 'netstandard, Version=2.0.0.0, Culture=neutral, PublicKeyToken=cc7b13ffcd2dd51'
```

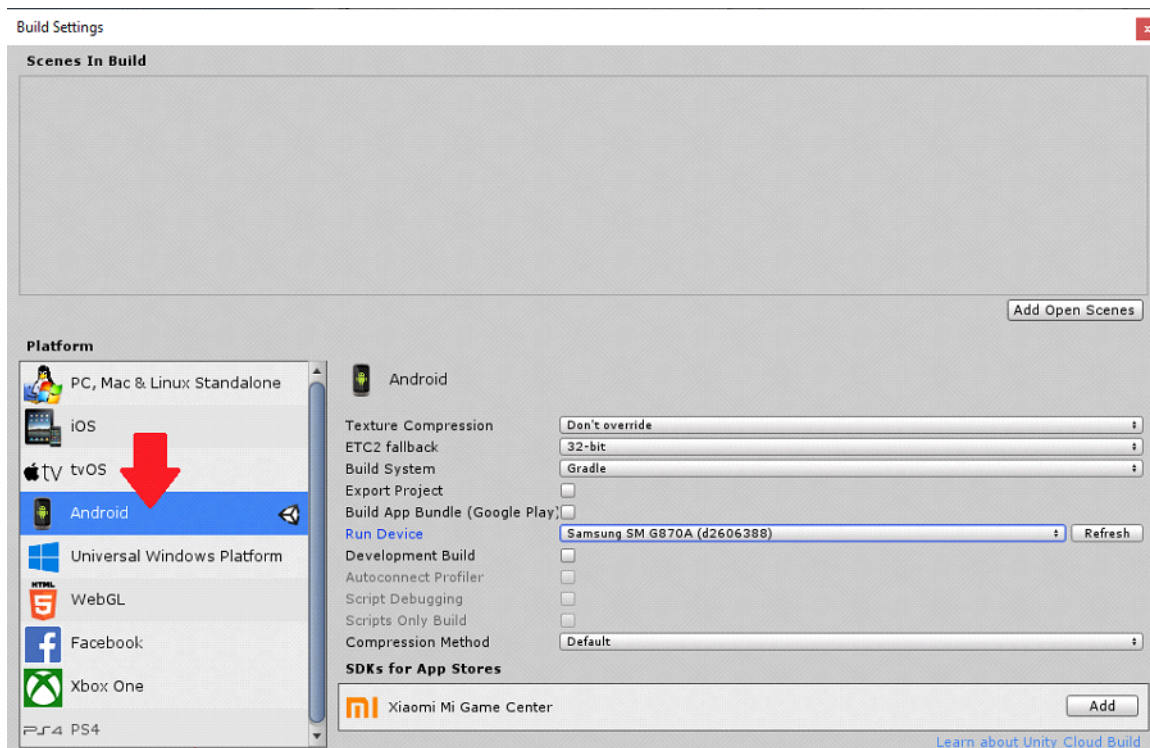
You will then get popups telling you that files are obsolete, say yes to deleting them.



That should be good for conflicts.

2. Be sure to now take the step to change the platform of your app if you haven't already done so to Android.

go to File > Build Settings, and change the platform to Android.



3. Make sure to setup braincloud like you usually would, where you go to the braincloud > Select Settings, and login and hook up your app on braincloud.

4. Now we can finally integrate and follow the Google Play Games Plugin Documentation to properly setup your app in Unity. <https://github.com/playgameservices/play-games-plugin-for-unity>. Unless you want the extra stuff, **ONLY DO THE CONFIGURE YOUR GAME** section of the documentation.

You may find that you need a resources definition, the easiest way to get this is to add an arbitrary achievement on google play. Also, **NOTE**: the Web Client ID you need is the one that was generated on your Google Cloud Platform project.

Google Play Games - Android Configuration

To configure Google Play Games in this project, go to the Play Game console, then enter the information below and click on the Setup button.  
[Open Play Games Console](#)

**Constants class name**  
Enter the fully qualified name of the class to create containing the constants

Directory to save constants: Assets  
Constants class name: GPGSIDs

**Resources Definition**  
Paste in the Android Resources from the Play Console

**Web App Client ID (Optional)**  
The web app client ID is needed to access the user's ID token and call other APIs on behalf of the user. It is not required for Game Services. Enter your oauth2 client ID below.  
To obtain this ID, generate a web linked app in Developer Console. Example: 123456789012-abcdefghijklmnopqrstuvwxyz.googleusercontent.com

Client ID:

Setup Cancel

Feature analytics

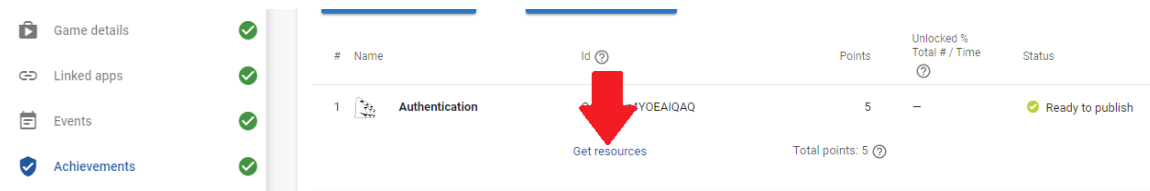
- Game details
- Linked apps
- Events
- Achievements**

Achievements are a fun way to encourage users to explore the game thoroughly and develop their mastery of it. They can be used to represent a user's accomplishments, such as beating a number of opponents or finishing a set of levels, or can represent a capability in the game that was not initially available. Achievements can be added on a regular basis to keep the game fresh and maintain users' engagement.

Learn all about implementing achievements in [Developer documentation](#).

**ADD ACHIEVEMENT** or **CONTINUE TO NEXT STEP**

Once added, with a picture logo you can get the resources.

#	Name	Id	Points	Unlocked % Total # / Time	Status
1	 Authentication	Y0EAIQAAQ	5	—	Ready to publish

Total points: 5

[Get resources](#)

## Export Resources

ANDROID

OBJECTIVE-C

JAVASCRIPT

TEXT

```
<?xml version="1.0" encoding="utf-8"?>
<!--
Google Play game services IDs.
Save this file as res/values/games-ids.xml in your project.
-->
<resources>
  <!-- app_id -->
  <string name="app_id" translatable="false">com.bitheads.716</string>
  <!-- package_name -->
  <string name="package_name"
translatable="false">com.bitheads.OauthGooglePlay</string>
  <!-- achievement Authentication -->
  <string name="achievement_authentication"
translatable="false">com.bitheads.716</string>
</resources>
```

DONE

5. After Google Play Games is setup, We can finally start to add the necessary code for authenticating with google and then with a google account to braincloud.

### Key snippets of code:

```
using Google;
```

This gives you access to Google's services. You may also want "using System.Threading.Tasks;" because we will use a thread for the callback.

```
GoogleSignInConfiguration configuration;
```

This is the configuration of your authentication. You can customize fields to obtain important codes and information about the account that is authenticated to google. The email and the IdToken are what is needed in order to authenticate a google account with Braincloud. You can initialize it and set the

values.

```
configuration = new GoogleSignInConfiguration
{
    WebClientId = webClientId,

    RequestEmail = true,

    RequestIdToken = true,

    RequestAuthCode = true
};
```

Note that this can also be set on the fly instead of on Start or Awake. You just need to access the Configuration of your google instance like this:

```
GoogleSignIn.Configuration.RequestEmail = true;

GoogleSignIn.Configuration.RequestIdToken = true;

GoogleSignIn.Configuration.RequestAuthCode = true;
```

As an example you may have a function and callback like this:

```
public void OnGoogleSignIn()
{
    //set the google configuration to the configuration object you set up
    GoogleSignIn.Configuration = configuration;

    //Can define this if you're using the game signin which we are.
    GoogleSignIn.Configuration.UseGameSignIn = true;

    //Can also define these tags here like this.
    //GoogleSignIn.Configuration.RequestEmail = true;
    //GoogleSignIn.Configuration.RequestIdToken = true;
    //GoogleSignIn.Configuration.RequestAuthCode = true;
```

```

        // With the configuration set, its now time to start trying to sign in.
        Pass in a callback to wait for success.

        GoogleSignIn.DefaultInstance.SignIn().ContinueWith(OnGoogleAuthSignIn);
    }

    //use a callback with a task to easily get results from the callback in order to get
    the values you need.

    public void OnGoogleAuthSignIn(Task<GoogleSignInUser> task)
    {
        if (task.IsFaulted)
        {
        }
        else
        {
            authCode = task.Result.AuthCode;

            idToken = task.Result.IdToken;

            email = task.Result.Email;
        }
    }
}

```

6. This whole process of signing in with Google with Google Sign In plugin will give you the Auth code for your app. This is essential because the Google Play games plugin mostly returns an empty string or null value for an auth code EVEN though you request it. Doing this method will confirm that you receive the auth code needed as this is needed for our braincloud server.

7. Now you will want to do some things in code to sign in with Google play games plugin. Doing this will return the google userId you will need to pass into braincloud for google authentication. The code will look something like this :

```

PlayGamesClientConfiguration config = new PlayGamesClientConfiguration.Builder()
    .RequestIdToken()
    .RequestServerAuthCode(false)
    .Build();

```

```

PlayGamesPlatform.InitializeInstance(config);

PlayGamesPlatform.Activate();

Social.localUser.Authenticate((bool success) => {

    if (success)
    {
        googleId = PlayGamesPlatform.Instance.GetUserId();
    }
    else
    {
    }
});

```

\*\*\* You may notice there is the ability to RequestServerAuthCode(), but you will only likely get an empty string with PlayGamesPlatform.Instance.GetServerAuthCode(); so that's why we need to use the google sign in plugin as well. \*\*\*

8. Finally, you can now use the GoogleId you stored, as well as the server auth code you stored from the google authentication you did with the other plugin and put those into the braincloud call to properly authenticate google using our old method.

```

AuthenticateGoogle(googleId, serverAuthCode, true, OnSuccess_Authenticate,
OnError_Authenticate);

```