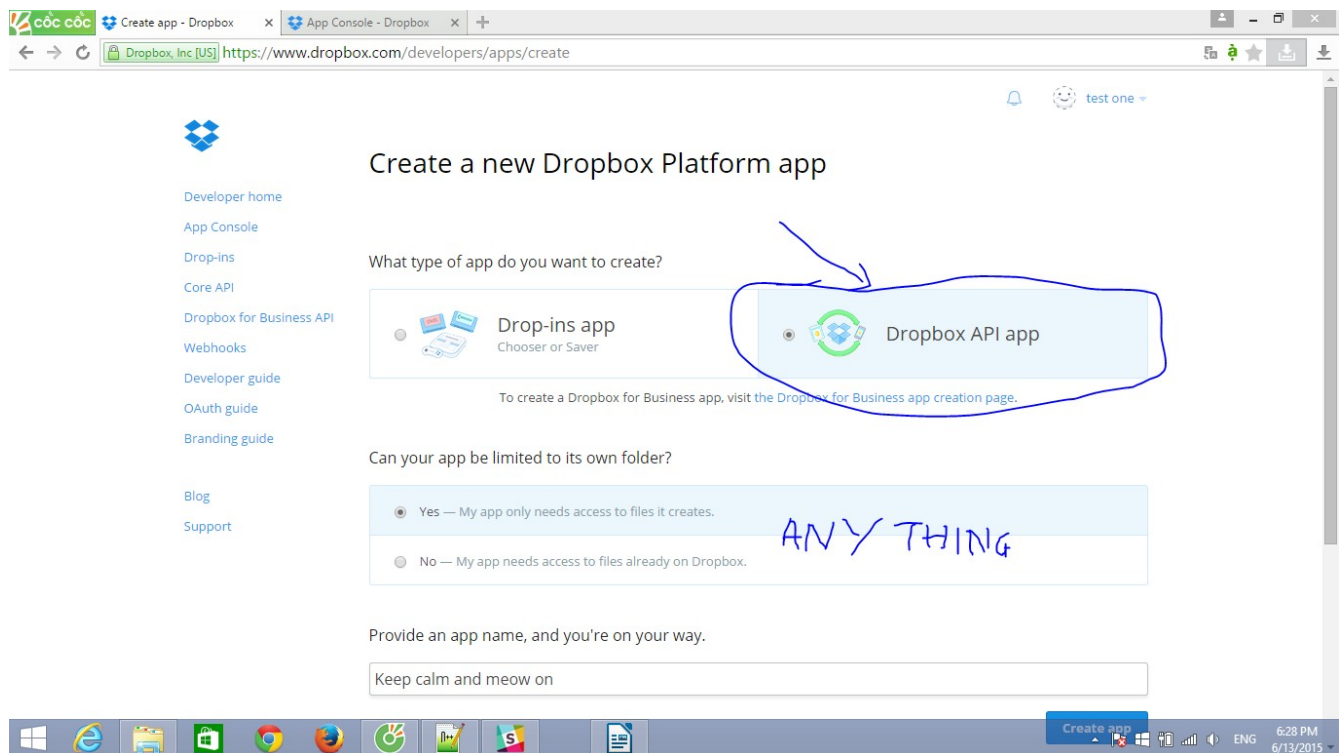
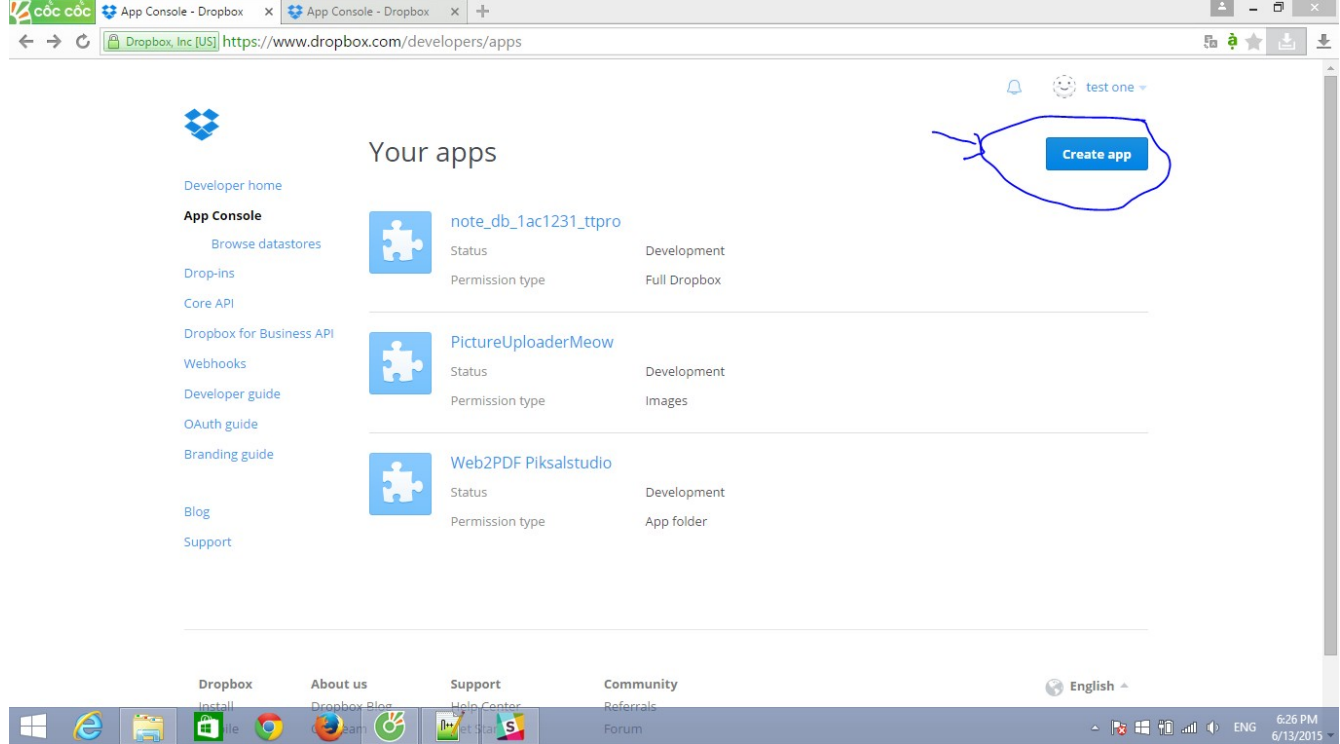
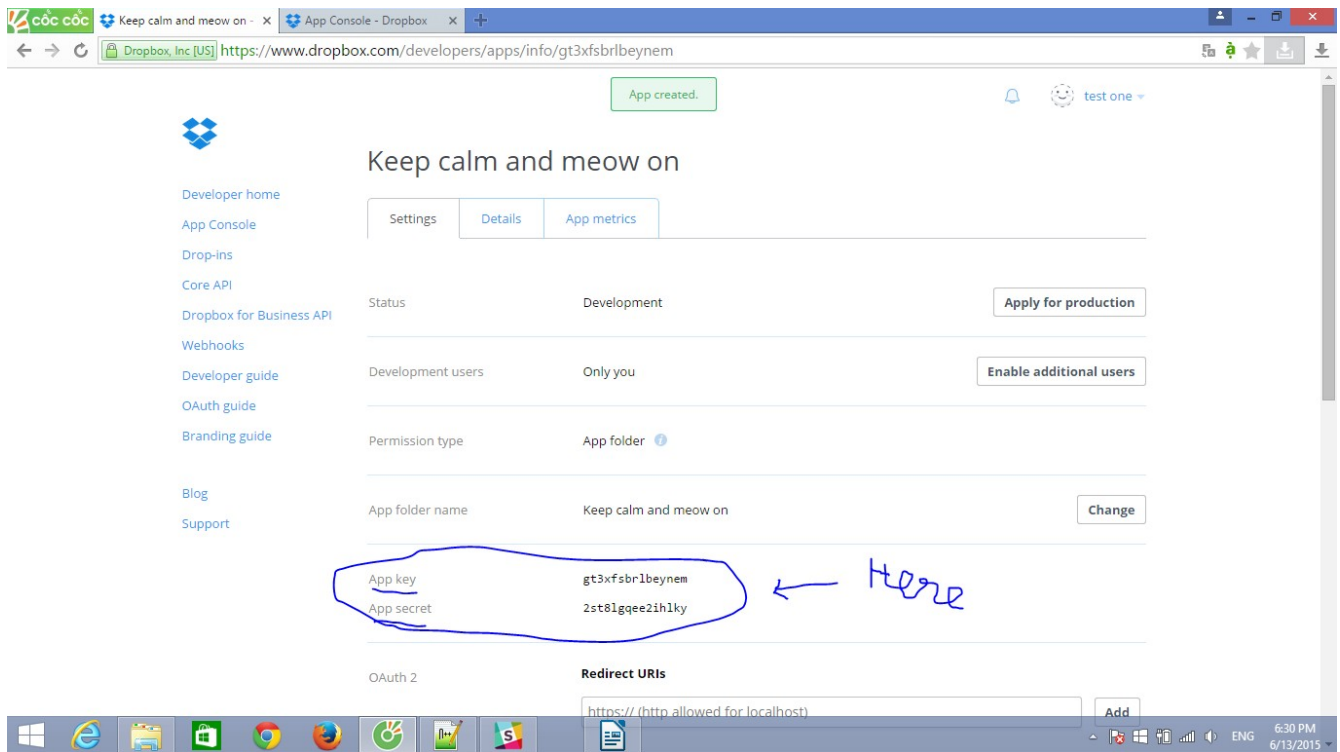


****CREATE APP in developer console**

Dropbox

Create new app in <https://www.dropbox.com/developers/apps> and get App_id, App_secret

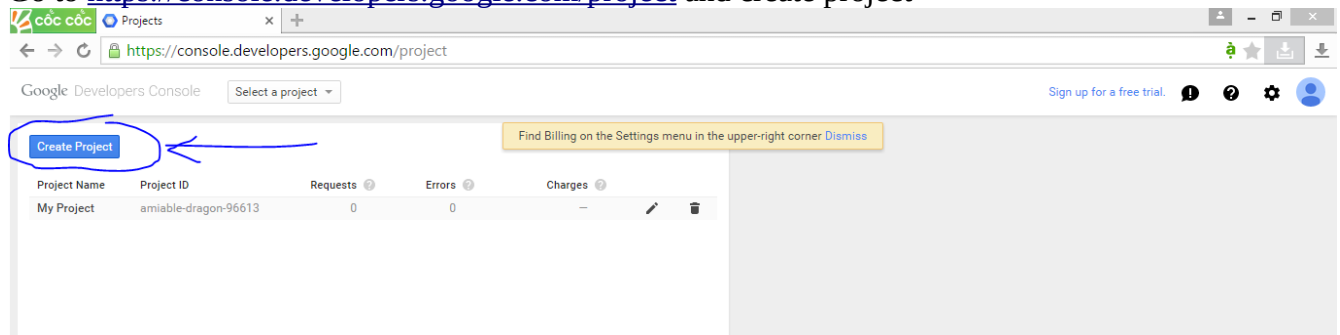




Save App key and App secret somewhere. You will need it.

Google Drive

Go to <https://console.developers.google.com/project> and create project



Enter project name

New Project

Project name ?

Keep Calm And Meow On

Your project ID will be keep-calm-and-meow-on ? Edit

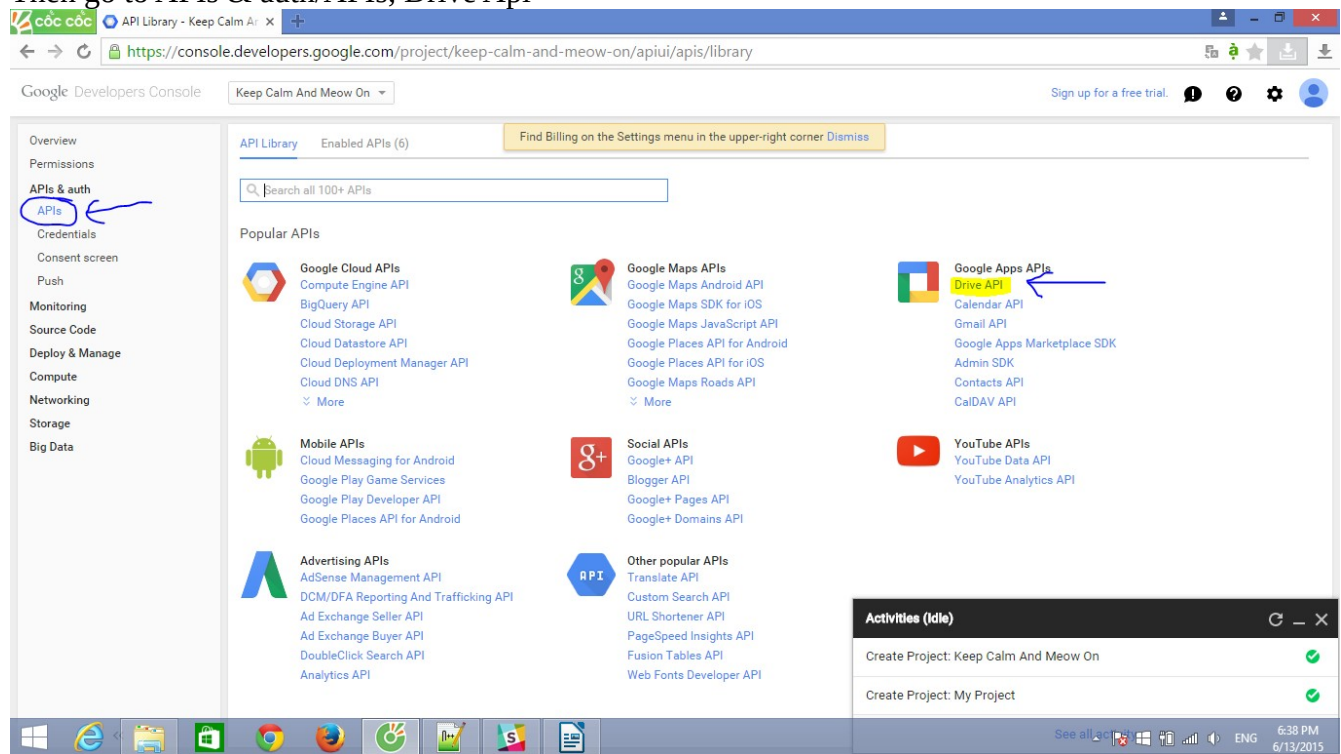
Show advanced options...

Create

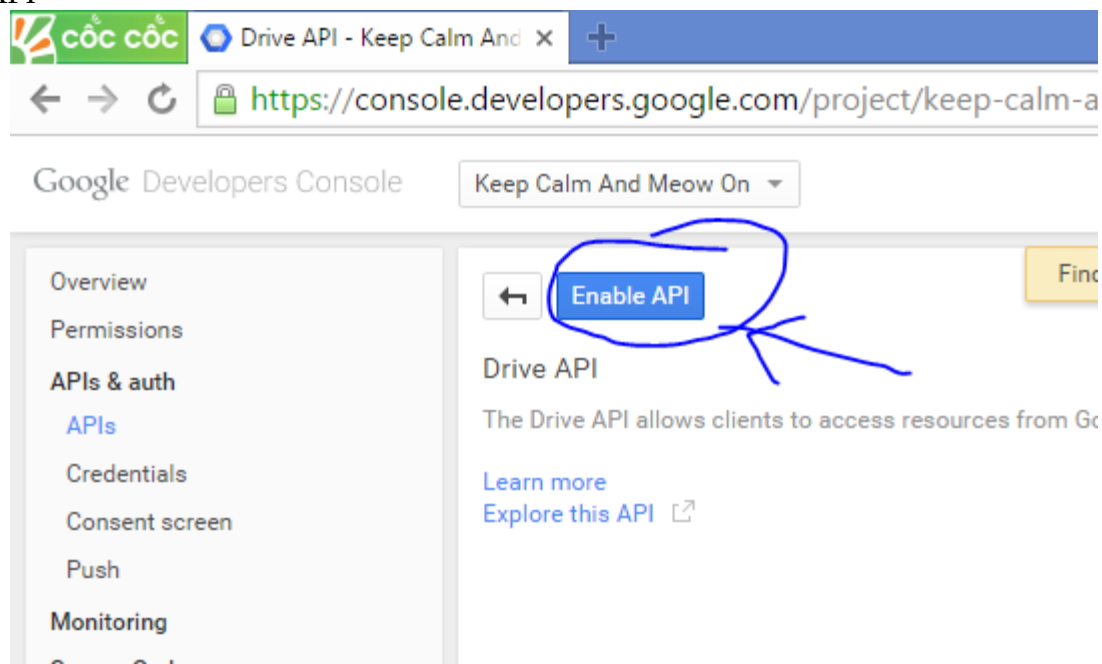
Cancel

Wait until project is created.

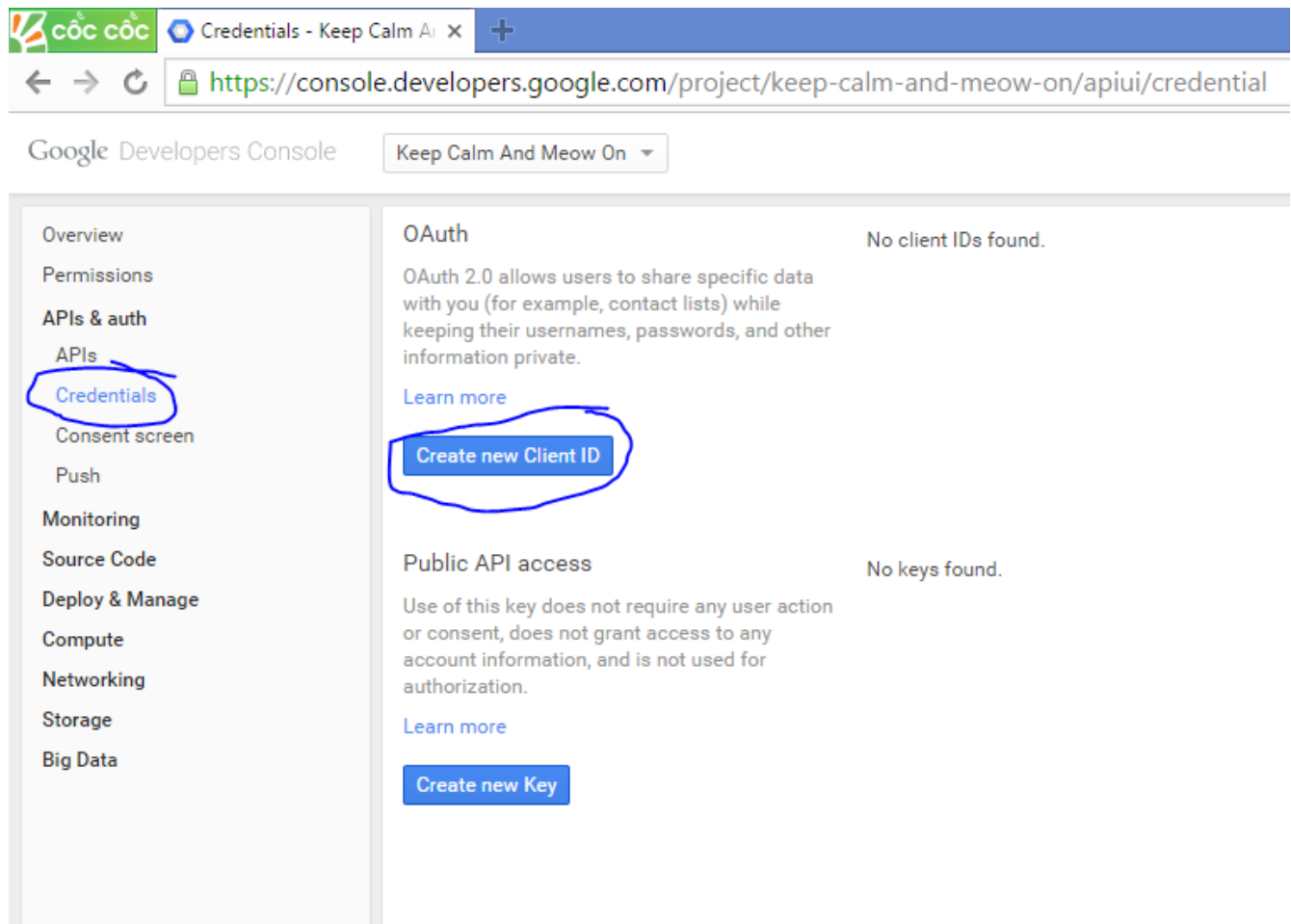
Then go to APIs & auth/APIs, Drive Api



Enable API




APIs & auth/ Credential and create new Client ID



Create Client ID

Application type

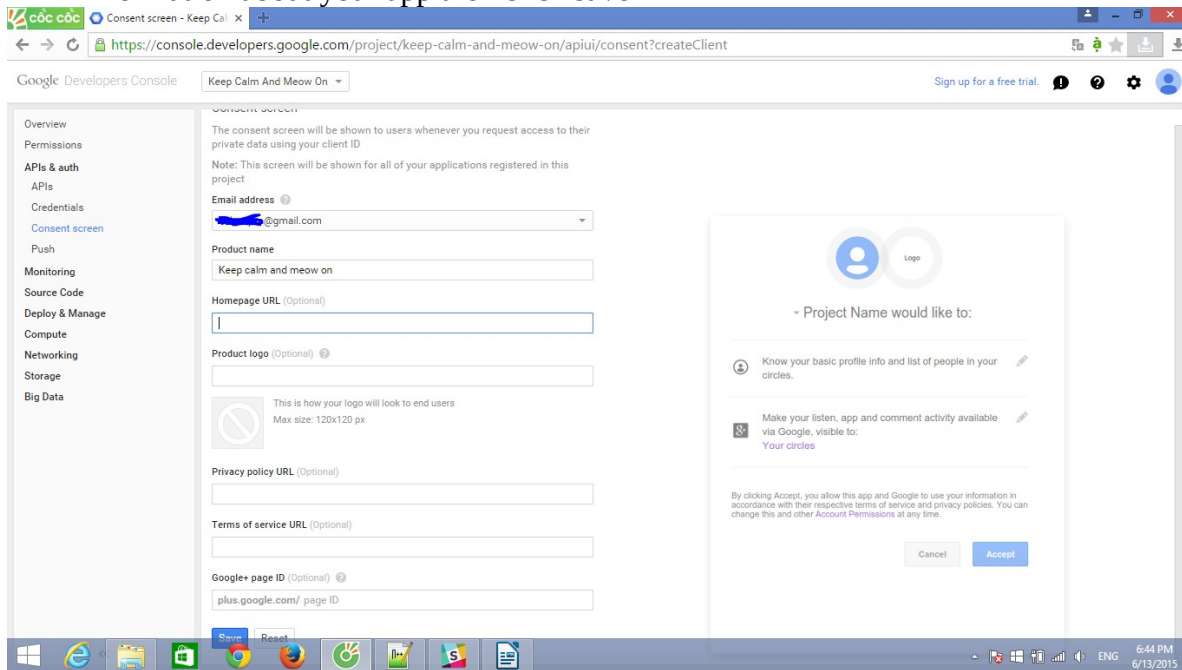
- ☐ Web application
Accessed by web browsers over a network.
- ☐ Service account
Calls Google APIs on behalf of your application instead of an end-user. [Learn more](#)
- ☒ Installed application
Runs on a desktop computer or handheld device (like Android or iPhone).

 To create a Web Client ID or an Installed Application Client, you need to set a product name in the consent screen.

[Configure consent screen](#)

[Cancel](#)

Fill in information about your app then click save



Now it is time to get SHA1 of your .keystore

+ find your debug.keystore

(debug.keystore is usually in "C:\Users\Your_User_Name\.android\debug.keystore")

+open CMD

+ navigate to C:\Program Files\Java\jdk1.8.0_45\bin

+ run command

keytool -exportcert -alias androiddebugkey -keystore "C:\Users\Thien\.android\debug.keystore" -list -v

+enter password: android

+ copy your SHA1

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Thien>cd "C:\Program Files\Java\jdk1.8.0_45\bin"

C:\Program Files\Java\jdk1.8.0_45\bin>keytool -exportcert -alias androiddebugkey
-keystore "C:\Users\Thien\.android\debug.keystore" -list -v
Enter keystore password:
Alias name: androiddebugkey
Creation date: Jun 4, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=Android Debug, O=Android, C=US
Issuer: CN=Android Debug, O=Android, C=US
Serial number: 78525c36
Valid from: Thu Jun 04 08:02:38 ICT 2015 until: Sat May 27 08:02:38 ICT 2045
Certificate fingerprints:
   MD5:  94:5E:03:F5:B0:58:12:C2:DA:09:59:DF:6F:07:4E:8E
   SHA1: 1C:B8:7D:4E:EB:6B:63:DD:93:E2:34:8F:10:5F:6E:91:26:5E:B2:B1
   SHA256: 75:79:E0:D0:71:17:36:7E:44:20:C9:2A:4D:15:F3:5D:B4:2A:4E:37:DA:
F6:1C:4A:3E:DE:23:57:C9:20:3F:B5
Signature algorithm name: SHA256withRSA
Version: 3

Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 30 76 D1 11 25 E4 FA CD  6E 70 F3 ED F7 A3 7D C9  00...np.....
0010: 95 23 B1 56                .#.U
]
]

C:\Program Files\Java\jdk1.8.0_45\bin>
```

+ Now fill in Create Client ID form

Create Client ID

Application type

☐ Web application
Accessed by web browsers over a network.

☐ Service account
Calls Google APIs on behalf of your application instead of an end-user. [Learn more](#)

☒ Installed application
Runs on a desktop computer or handheld device (like Android or iPhone).

Installed application type

☒ Android [Learn more](#)

☐ Chrome Application [Learn more](#)

☐ iOS [Learn more](#)

☐ PlayStation 4

☐ Other

API requests are sent directly to Google from your clients' Android devices. Google verifies that each request originates from an Android application that matches the package name and SHA1 signing certificate fingerprint name listed below.

Package name

com.meow.keep_calm_and_meow_on

Signing certificate fingerprint (SHA1)

1C:B8:7D:4E:EB:6B:63:DD:93:E2:34:8F:10:5F:6E:91:26:5E:B2:B1

Deep linking

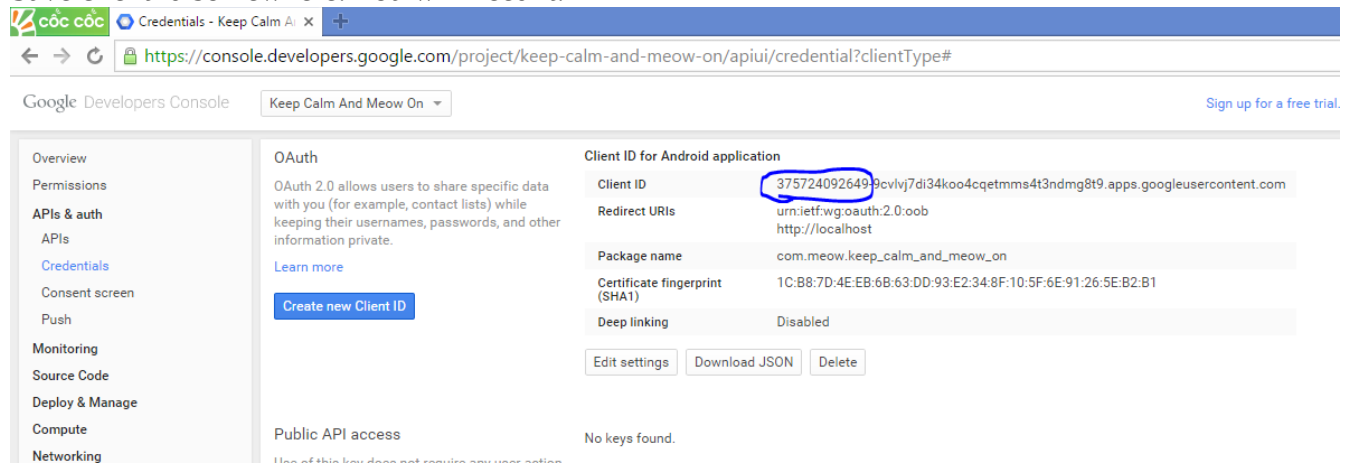
☐ Enabled

☒ Disabled

Create Client ID

Cancel

Save client id somewhere. You will need it.



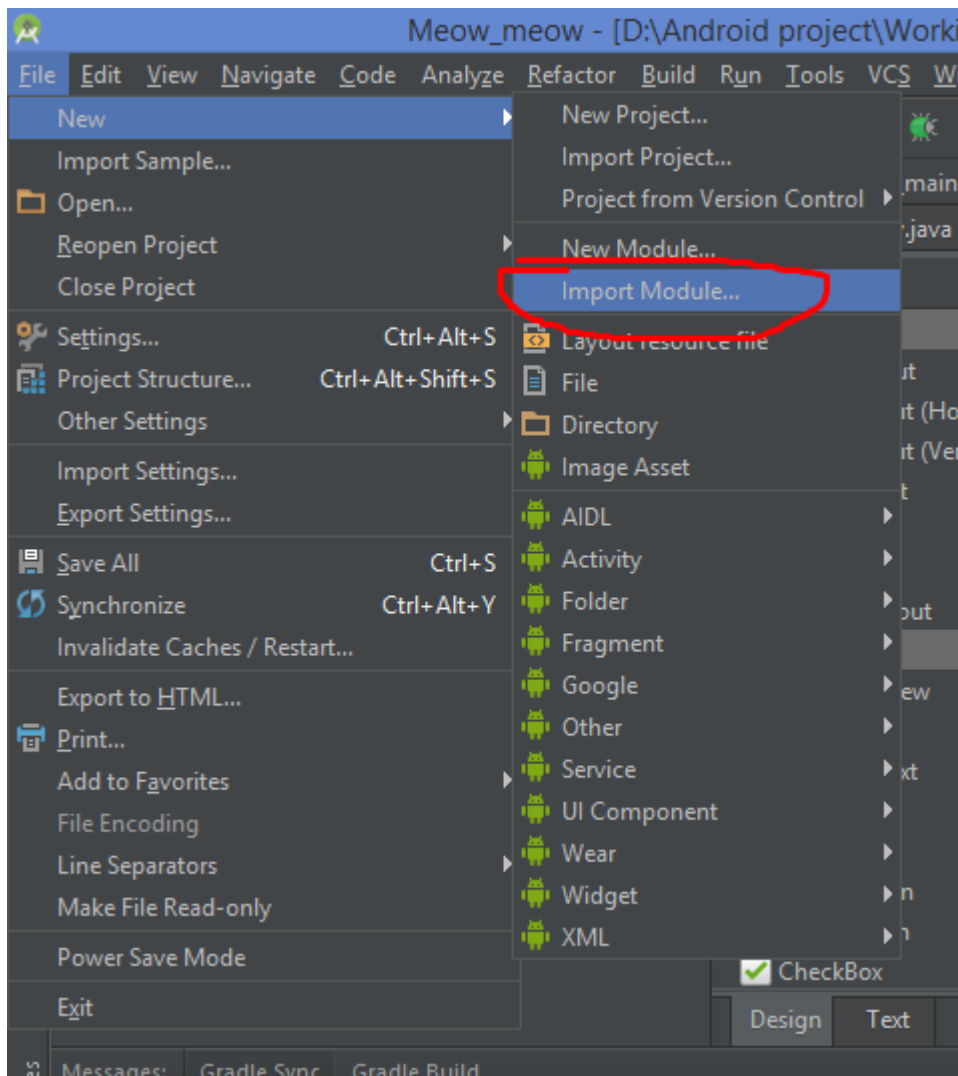
The screenshot shows the Google Developers Console interface for a project named "Keep Calm And Meow On". The left sidebar contains navigation links: Overview, Permissions, APIs & auth (selected), Credentials, Consent screen, Push, Monitoring, Source Code, Deploy & Manage, Compute, and Networking. The main content area is divided into two sections. The top section, "OAuth", provides information about OAuth 2.0 and includes a "Create new Client ID" button. The bottom section, "Public API access", states "No keys found." The right section, "Client ID for Android application", displays the following details:

Client ID for Android application	
Client ID	375724092649-9cvtvj7di34koo4cqetmms4t3ndmg8t9.apps.googleusercontent.com
Redirect URIs	urn:ietf:wg:oauth:2.0:oob http://localhost
Package name	com.meow.keep_calm_and_meow_on
Certificate fingerprint (SHA1)	1C:B8:7D:4E:EB:6B:63:DD:93:E2:34:8F:10:5F:6E:91:26:5E:B2:B1
Deep linking	Disabled

Below the table are three buttons: "Edit settings", "Download JSON", and "Delete".

IMPORT MODULE TO PROJECT

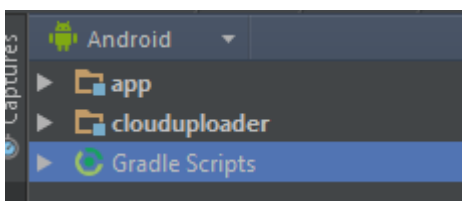
Import module



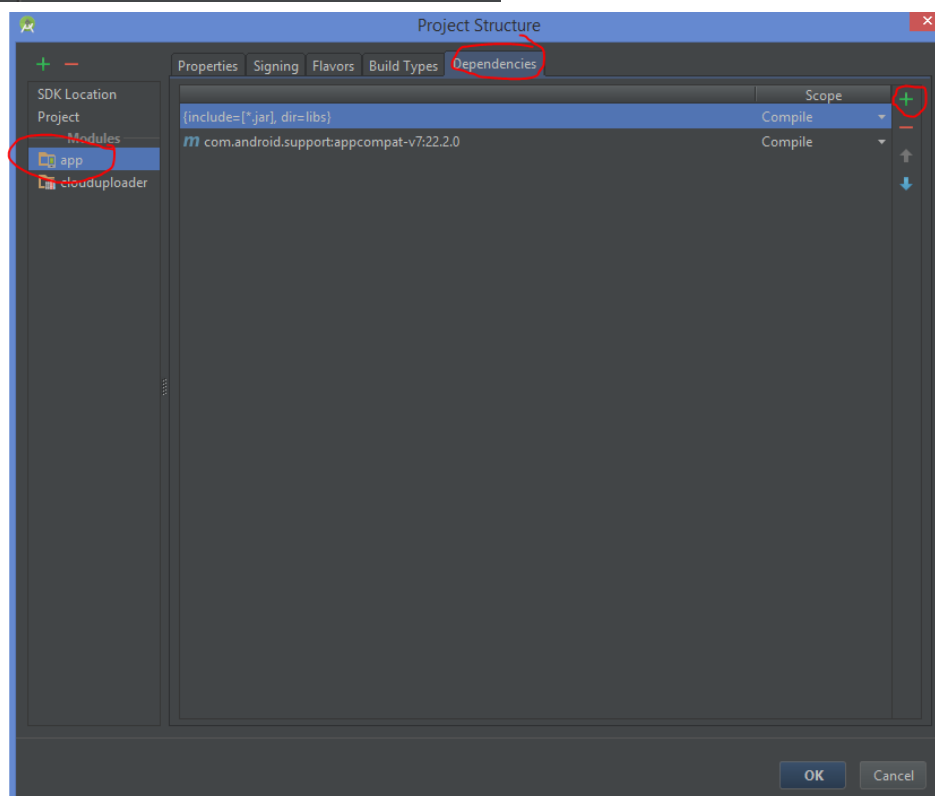
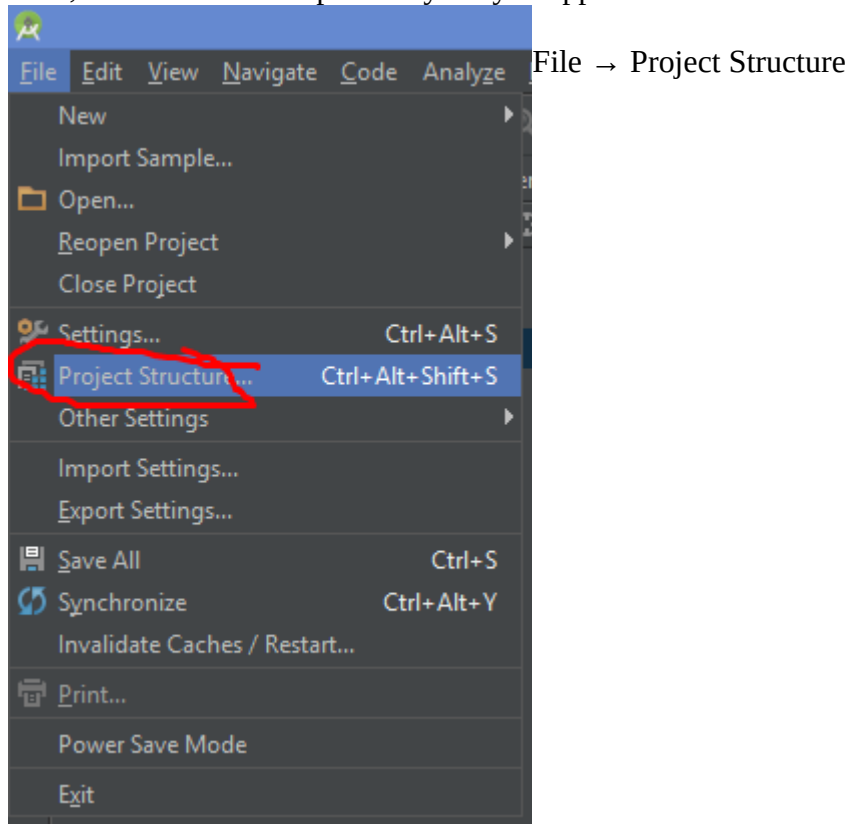
From ...\\PictureUploader2\\clouduploader

Wait android studio import for you. It take about 1 to 5 minutes.

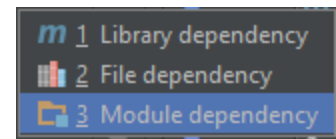
You will see like this



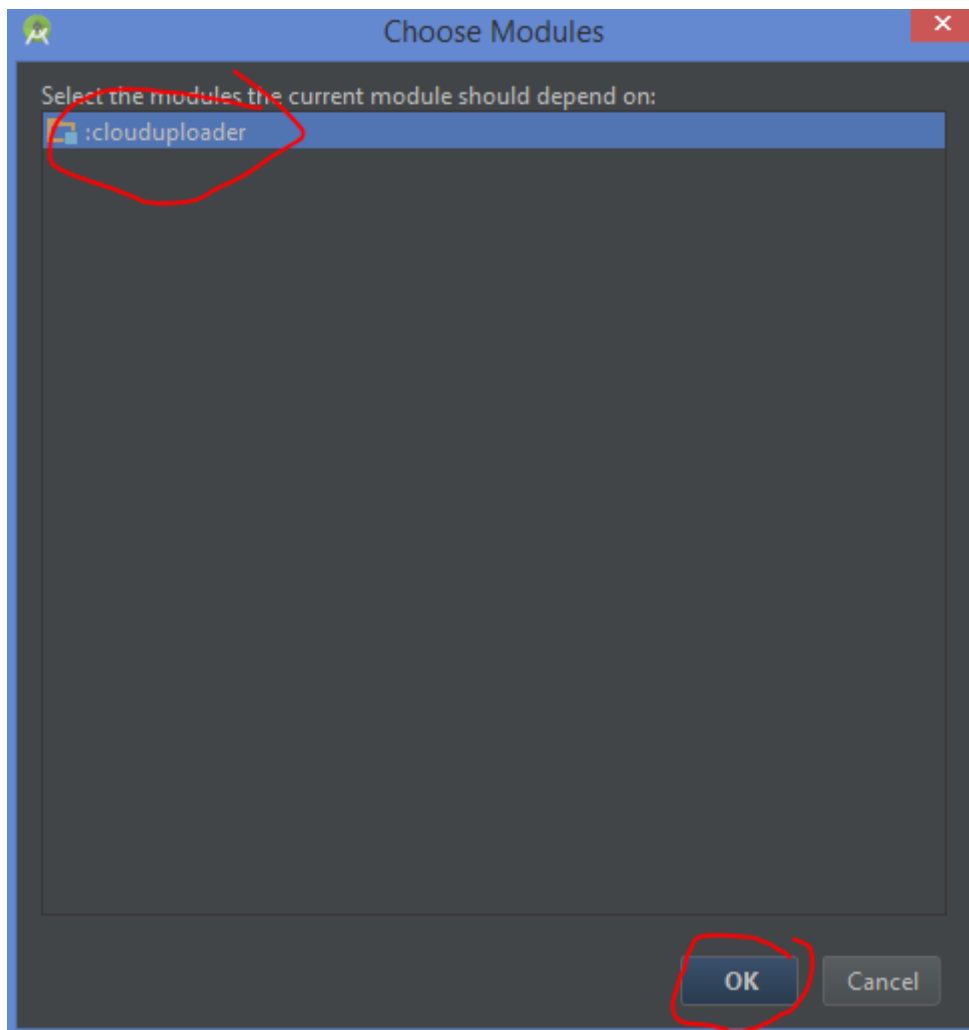
Next, we need to add dependency for you app.



In project Structure form. On Modules (left panel), choose “app”. Then choose tag Dependencies. Then click “+”



Choose Module dependency



Choose clouduploader and click ok.

Wait a bit for android studio to work for you.
Now you are done import clouduploader to your project.

SET UP DROPBOX AND GOOGLE DRIVE IN YOUR PROJECT

In manifest, add some permission.

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

Add login activity, for both dropbox and google drive

```
<activity
    android:name="com.PiksalStudio.thien.clouduploader.LoginActivity"
    android:label="@string/title_activity_login"
    android:parentActivityName=".MainActivity" >
    <meta-data
        android:name="com.google.android.apps.drive.APP_ID"
        android:value="id=1025765906493" />
    <intent-filter>
        <action android:name="com.google.android.apps.drive.DRIVE_OPEN" />
        <data android:mimeType="application/vnd.google-apps.drive-sdk.1025765906493" />
        <data android:mimeType="image/png" />
        <data android:mimeType="image/jpeg" />
        <data android:mimeType="image/jpg" />
    </intent-filter>
</activity>
```

Replace 1025765906493 with your google client ID

Add dropbox activity

```
<activity
    android:name="com.dropbox.client2.android.AuthActivity"
    android:configChanges="orientation|keyboard"
    android:launchMode="singleTask" >
    <intent-filter>
        <!-- Change this to be db- followed by your app key -->
        <data android:scheme="db-gg5bv3ify8ok6td" />
        <action android:name="android.intent.action.VIEW" />
        <category android:name="android.intent.category.BROWSABLE" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

Replace gg5bv3ify8ok6td with your app key.

For ever activities that use google drive, add

```
<meta-data
    android:name="com.google.android.apps.drive.APP_ID"
    android:value="id=1025765906493" />
and
<intent-filter>
<action android:name="com.google.android.apps.drive.DRIVE_OPEN" />

    <data android:mimeType="application/vnd.google-apps.drive-sdk.1025765906493" />
    <data android:mimeType="image/png" />
    <data android:mimeType="image/jpeg" />
    <data android:mimeType="image/jpg" />
</intent-filter>
```

Remember to replace 1025765906493 with your google client ID.

Initialize cloudu uploader by call it public constructor.

```
public CloudUploader( Activity activity, String APP_FOLDER_NAME, String Dropbox_app_id,
String Dropbox_app_secret)
```

Activity: the current activity, which we initialize cloudu uploader. (ex: My_Activity.class)

APP_FOLDER_NAME: folder with file will be upload to. (ex: my_picture)

Dropbox_app_id and Dropbox_app_secret : Dropbox app key and dropbox app secret we already got from app console.

LOGIN

```
public void StartLoginActivity()
```

It will start Login Activity, which allow user to login dropbox and google drive. Call it after constructor. But you may only need to call it once for whole app, not every instance of CloudUploader, because login token are saved in SharedPreferences.

You can customize layout in CloudUploader module.

"%\PictureUploader2\cloudu uploader\src\main\res\layout\activity_login.xml"

Upload To Dropbox

```
public void UploadFileDropbox(String Name, InputStream is, Long Length,Handler handler )
```

Name: name of file you want to store on dropbox. Including extension (ex: my_avata.png)

is: InputStream of file

Length: size of file.

File will be upload into folder named we have defined in APP_FOLDER_NAME

Handler : handler should be initialize in activity before pass in here .

```
dropbox_handler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (msg.arg1==1) {
            Log.i("DROPBOX HANDLER", "Upload ok");
            //do something when upload finished
        }
        if (msg.arg1==-1) {
            Log.i("DROPBOX HANDLER", "Upload failed");
            //do something when upload failed
        }
    }
};
```

Upload To Google Drive

public void UploadFileGoogleDrive(String Name, InputStream is, Long Length,Handler handler)

Name: name of file you want to store on dropbox. Including extension (ex: my_avata.png)

is: InputStream of file

Length: size of file.

File will be upload into folder named we have defined in APP_FOLDER_NAME

Handler : handler should be initialize in activity before pass in here .

```
google_drive_handler = new Handler(){
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if (msg.arg1==1) {
            Log.i("GOOGLE DRIVE HANDLER", "Upload ok");
            //do something when upload finished
        }
        if (msg.arg1==-1) {
            Log.i("GOOGLE DRIVE", "Upload failed");
            //do something when upload failed
        }
    }
};
```

Login To Dropbox

public void LoginDropbox()

Show web browser with login dropbox ui for user.

After user loggedin, ask for permission and back to current activity which called clouduploader.LoginDropbox()

Check if Dropbox is loggedin

public boolean Dropbox_isLogin()

Return true when user already loggedin succesfully.

Then you can upload file to dropbox .

This function have it own dummy activities which will be destroy when it login successfully, so there is nothing to do with your activity.

Change Google Drive Account

`public GoogleApiClient SelectGoogleAccount(int Google_API_request_code)`

Google_API_request_code: your own define request code which will be used in onActivityResult.

Return GoogleApiClient which you will use in onActivityResult.

In your activity which is called clouduploader.SelectGoogleAccount(your_request_code), you must edit manifest xml (see page 11) and have onActivityResult

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == Google_API_request_code)
        if (resultCode==RESULT_OK) {
            mGoogleApiClient.connect();
            Log.i("google","result ok");
        }
}
```

mGoogleApiClient is return value from

clouduploader.SelectGoogleAccount(your_request_code).

Google_API_request_code is your_request_code.

GET EXAMPLE AND MODULE HERE

<https://github.com/ttpro1995/PictureUploader2>