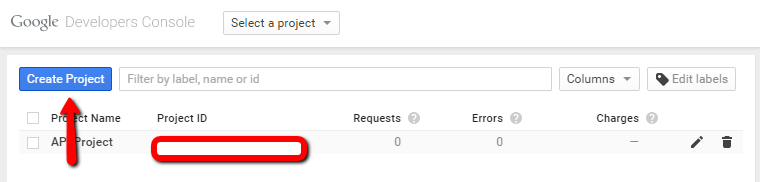
Day 26: Remote Notifications with Parse

Today, we will explore how to create Remote Notifications in Xamarin.Android. To do remote notifications in any Android Application, we need to work with Google Cloud Messaging (GCM). This process of working with GCM is a bit complicated and there are number of providers out there that take care of this complicated procedure for you and expose simple APIs for you to hook into. For this sample, we will take a look at how to use [Facebook’s Parse](http://parse.com) but you can use any Push Notification Provider there is. For example like Microsoft Azure.

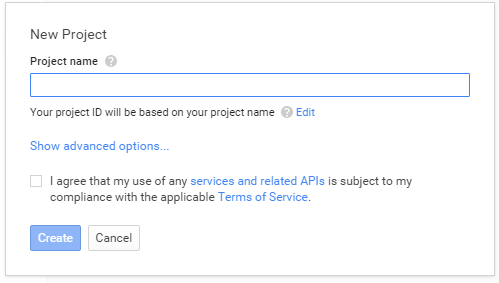
Let’s take a look at the steps to get Push Notifications working in our Xamarin.Android App –

# Configuration on Google Cloud Messaging

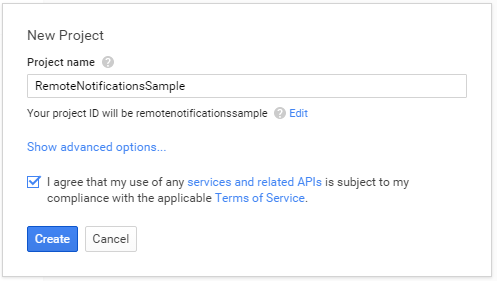
First go to the [Google Developers Console](http://cloud.google.com/console) and sign-in with your Google account. Once you are signed-in, you will be taken to the Dashboard for the Google Developers Console. On there, you should see a button to “Create Project”



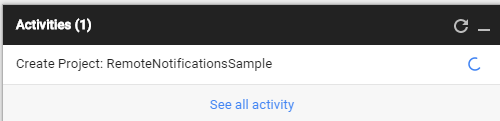
Once you click Create Project, you will be presented with the “New Project” Dialog which asks you for the Project Name.



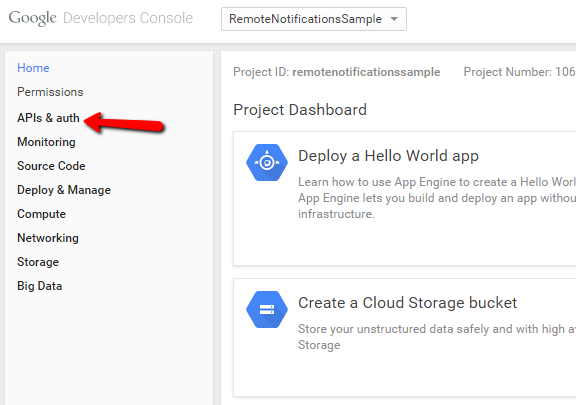
Fill in the project name and for this sample, let’s call the project “RemoteNotificationSample” and agree to the terms and click Create –



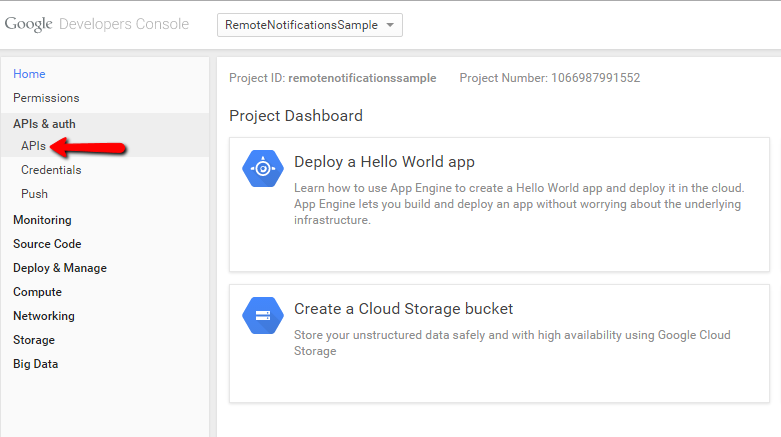
Once you click Create, Google would go and create the project for you. You may see a little Activities window on the bottom right showing the status of creation –

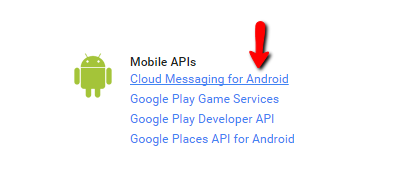


Once the project creation is successful, next step is to setup API Key for our Push Provider to interact with GCM. This can be done on the Project Dashboard and on the left navigation menu there is an option for “APIs & auth”. Go ahead and click it –

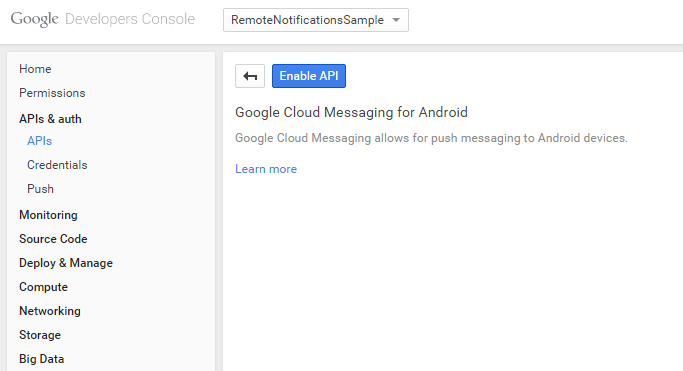


Underneath APIs and Auth, you will then see an APIs menu option, go ahead and click it –

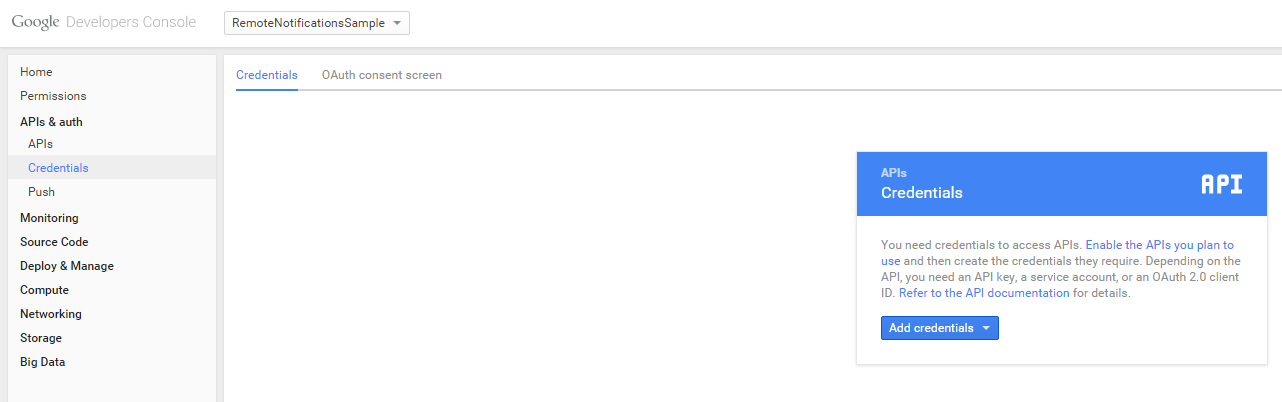


You will then be presented with bunch of APIs you can setup for your project and one of the section is “Mobile APIs”. Under this section, there is an option for “Cloud Messaging for Android”. Go ahead and click it –

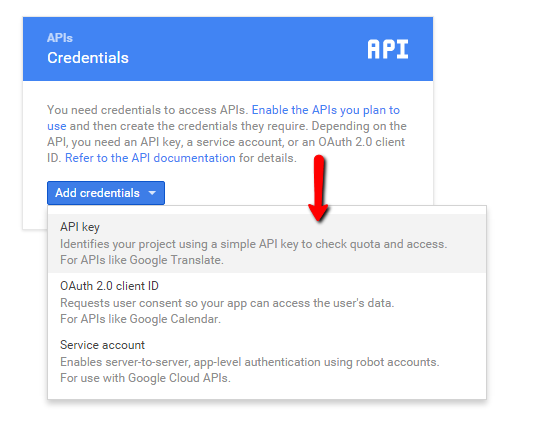
You will be presented with a little screen asking for your confirmation to enable Cloud Messaging for Android, click Enable API on this screen –



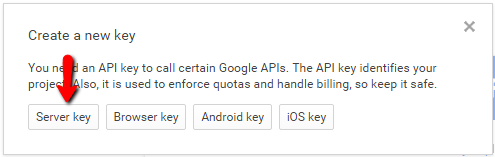
Once you click Enable API button, you will be then presented with API Credentials screen and an option to add credentials –



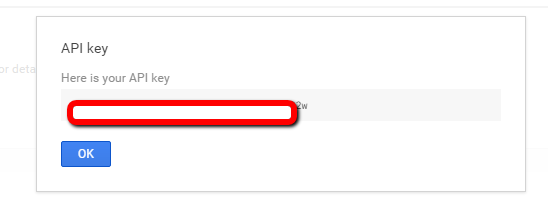
Click the little pull-down for “Add credentials” and click on “API Key” option.



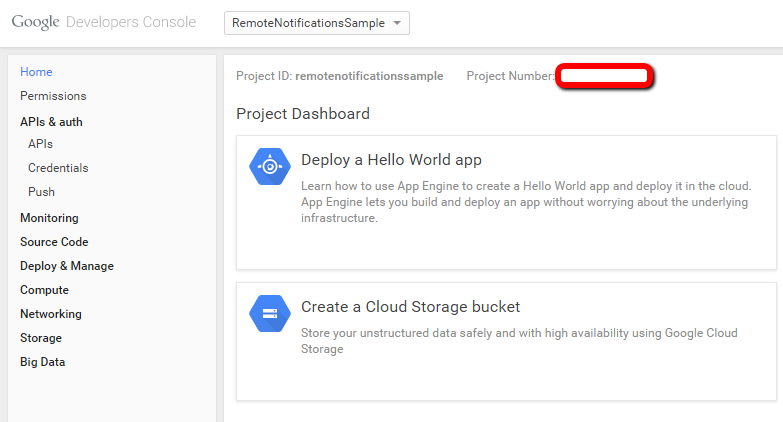
You will then be asked for create a new key and from the options presented, go ahead and click “Server Key” –



And finally, you will be presented with the API Key for your Android App. Take a note of this key –



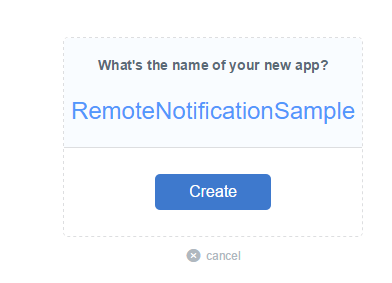
Also, on the Project Dashboard is the Project ID information, please take a note of this as well as we will be using Project ID in the Parse Portal –



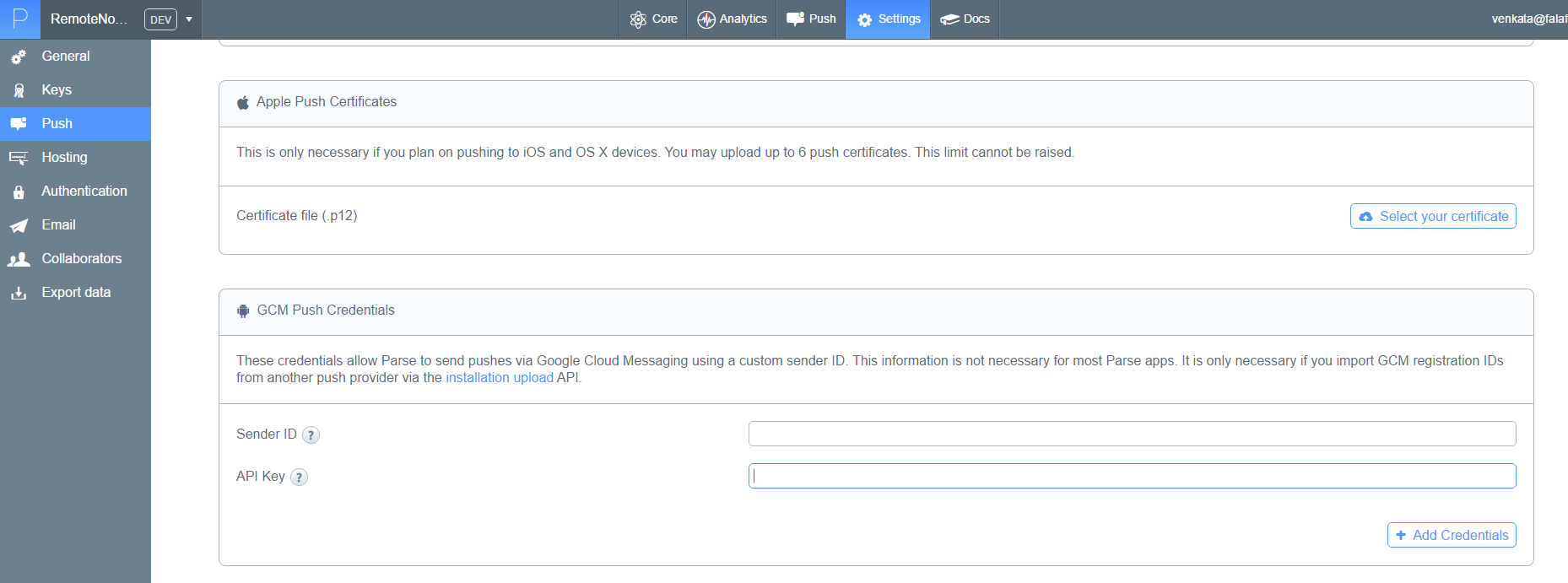
# Configuring Parse

Now that we have configured GCM, we now need to configure our Push Provider, Parse, and tell them about our API Key and ProjectID. Let’s take a look at how that is done –

On your Parse Dashboard click “Create a Project” and let’s call the project RemoteNotificationSample –



Once the project creation is successful, click on the “settings” button in the project dashboard menu and the click on “Push” from the left navigation menu –

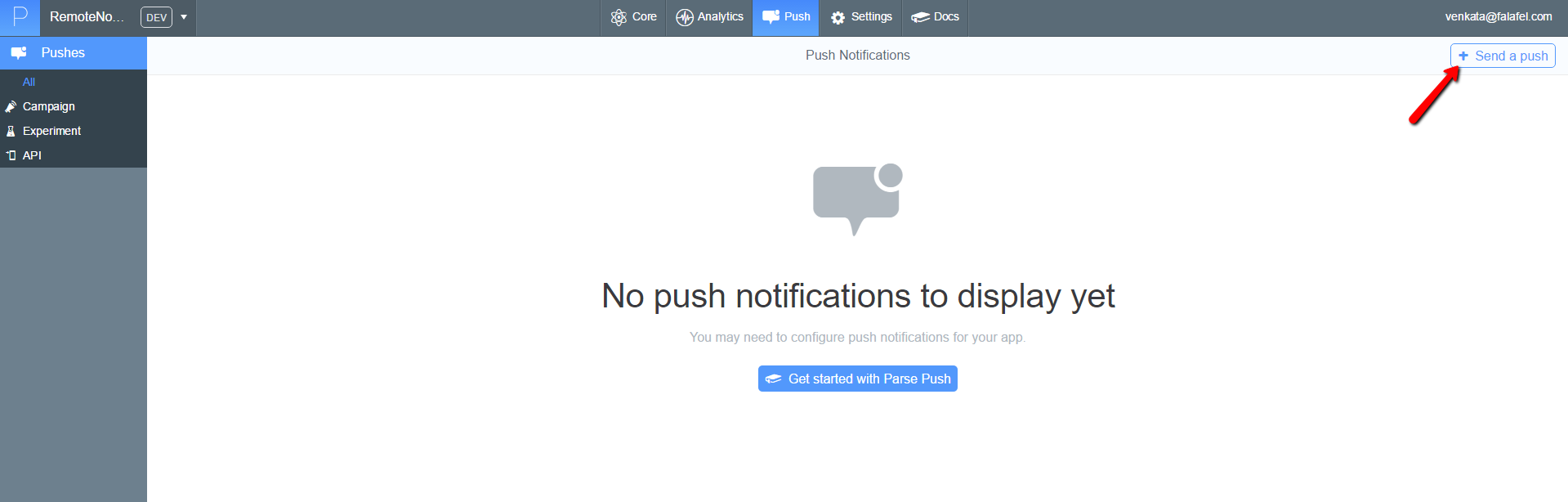


You will see a “GCM Push Credentials” section asking you for your Sender ID (ProjectID) and API Key(Key we got from the Developers Console). Enter those two and save them.

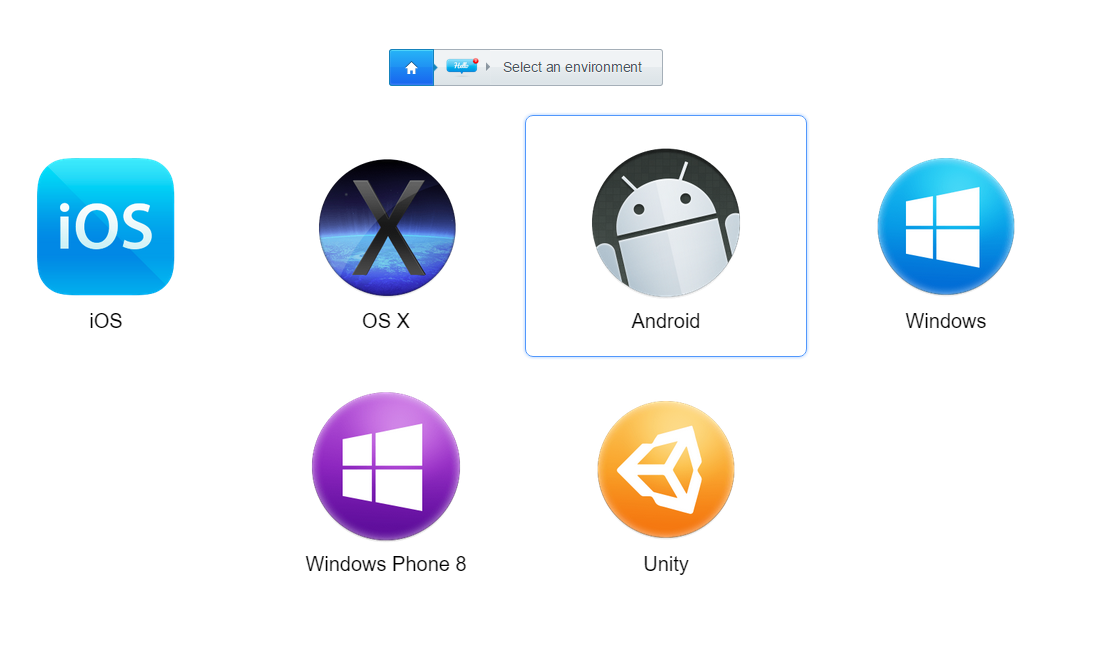
Now that we have told Parse about our GCM credentials, Parse will do the heavy lifting for us on communication with GCM. Let’s start the work on our Android App to allow it to receive notifications. Fortunately Parse has a really good “Getting Started” guide (and apps) to help us through this process.

# Configuration on Xamarin.Android

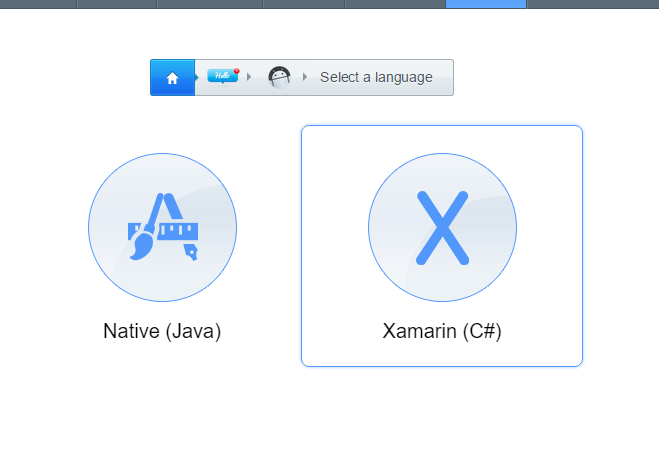
Let’s click the “Push” button on the Parse project menu, and click on “Get started with Parse Push” button –



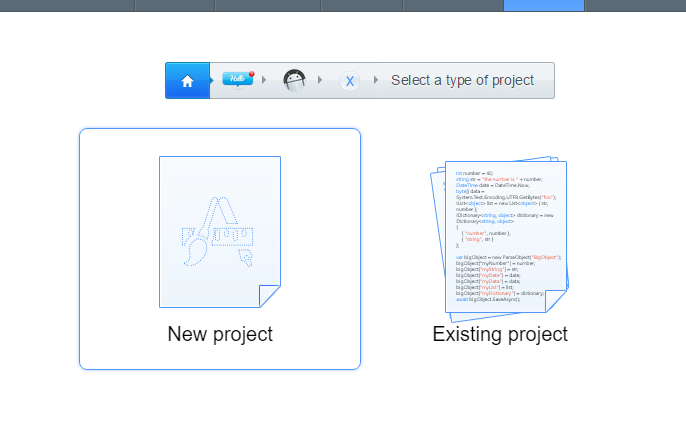
We will then be presented with a getting started guide page which lets us select our environment –



Lets select Android, and in Android, select Xamarin (C#)



Once you do that, Parse lets you download a full Xamarin project (either new or existing)



Let’s download the app and open it up and take a look at the important code –

Parse uses Parse’s SDK to do all interactions with Parse. The pre-built project already has the reference to “Parse.Android” set –

Our first task in the code is to open up **ParseApplication.cs** file and pass the right parameters for ParseClient.Initialize method –

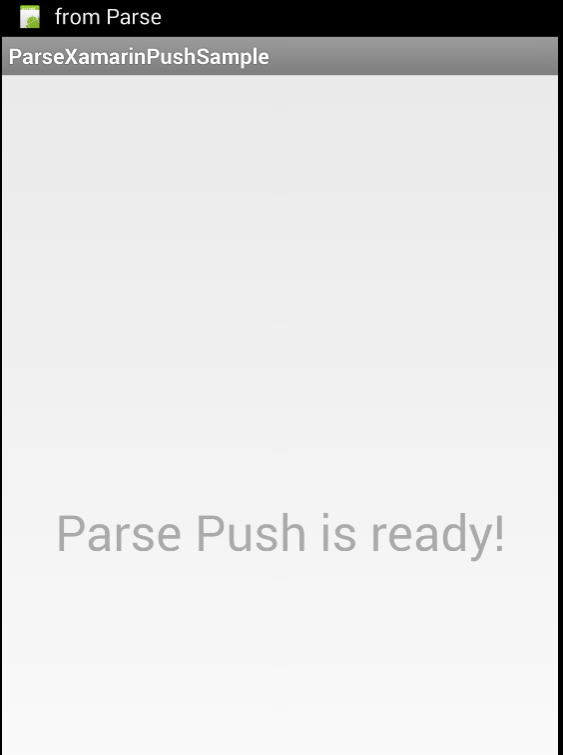
|  |
| --- |
| using System;  using Android.App;  using Android.Runtime;  using Parse;  namespace ParseXamarinPushSample  {  [Application(Name = "parsexamarinpushsample.ParseApplication")]  class ParseApplication : Application  {  public ParseApplication(IntPtr handle, JniHandleOwnership ownerShip)  : base(handle, ownerShip)  {  }  public override void OnCreate()  {  base.OnCreate();  ParseClient.Initialize("<APPLICATIONID>", "<.NETKEY>");  ParsePush.ParsePushNotificationReceived += ParsePush.DefaultParsePushNotificationReceivedHandler;  }  }  } |

Next thing to observe is the AndroidManifest.xml file –

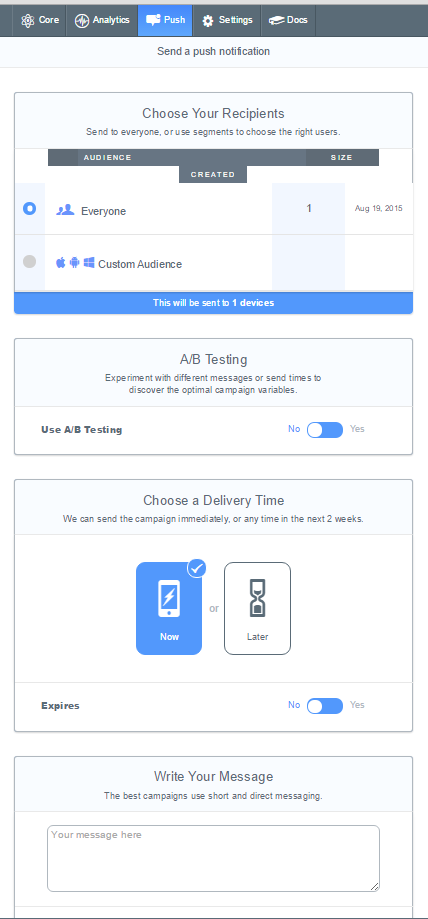
|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.parse.parsexamarinpushsample" android:versionCode="1" android:versionName="1.0" android:installLocation="auto">  <uses-sdk android:minSdkVersion="10" android:targetSdkVersion="21" />  <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  <uses-permission android:name="android.permission.WAKE\_LOCK" />  <uses-permission android:name="android.permission.VIBRATE" />  <uses-permission android:name="android.permission.GET\_ACCOUNTS" />  <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />  <permission android:protectionLevel="signature" android:name="com.parse.parsexamarinpushsample.permission.C2D\_MESSAGE" />  <uses-permission android:name="com.parse.parsexamarinpushsample.permission.C2D\_MESSAGE" />  <application android:name="parsexamarinpushsample.ParseApplication" android:label="ParseXamarinPushSample" android:icon="@drawable/Icon" android:theme="@android:style/Theme.Light">  <receiver android:name="parse.ParsePushBroadcastReceiver" android:permission="com.google.android.c2dm.permission.SEND">  <intent-filter>  <action android:name="com.google.android.c2dm.intent.RECEIVE" />  <action android:name="com.google.android.c2dm.intent.REGISTRATION" />  <category android:name="com.parse.parsexamarinpushsample" />  </intent-filter>  </receiver>  <service android:name="parse.ParsePushService" />  </application>  </manifest> |

Any application that accepts push needs some permissions, namely: INTERNET, ACCESS\_NETWORK\_STATE,WAKE\_LOCK,VIBRATE,GET\_ACCOUNTS, and C2DMRECEIVE. There are few more permissions that are required for push notifications, and all those are documented above.

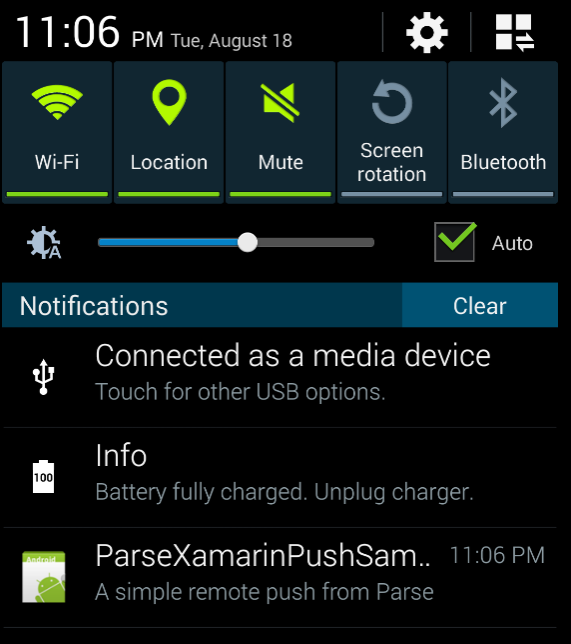
Finally, if you run the application, you should just see a simple start page like below –



Now, let’s head back to the Parse portal and send ourselves a test Push notification. You can do so by click “Send a Push” button. The send a push page allows us to choose recipients, do A/B testing, configure delivery time and more. Most importantly, it lets you put in the text we want the Push Notification to have –

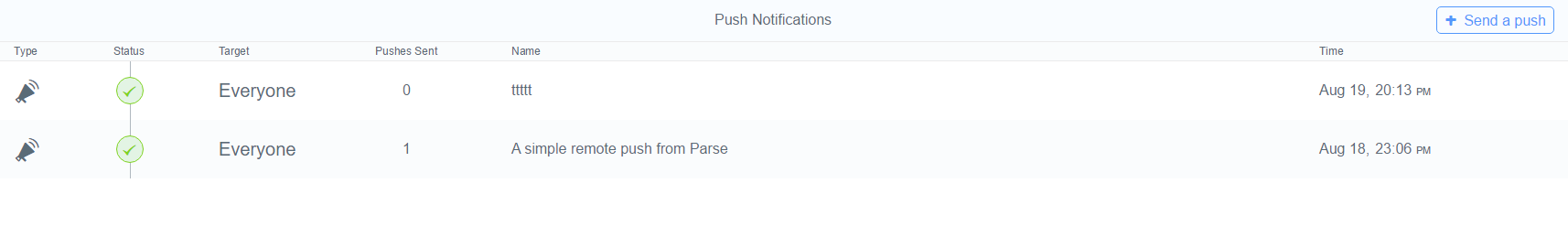


Fill in all the required fields and click “Send Now”. Your Android App should now receive a remote notification –



And voila, we have our first Push Notification for our Xamarin.Android application.

Also, in the Parse portal, you can see the status of your push notifications –



That’s it for today, see you all tomorrow.