

Nearme App

Documentation for Windows 7/8

1. Installation

- 1.1 Install Java
- 1.2 Install the Android SDK
- 1.3 Install Git
- 1.4 Install Node.js
- 1.5 Install Dependencies
- 1.6 Environment variables

2. Configuration

- 2.1 App configuration
- 2.2 Theme
- 2.3 Google Analytics
- 2.4 Push Notifications
 - 2.4.1 Configure PushWoosh
 - 2.4.2 Configure Push notifications for Android
- 2.5 AdMob
- 2.6 Google Maps
 - 2.6.1 Print certificate fingerprint
 - 2.6.2 Obtain Google Maps API Key for Android

3. Android Building

- 3.1 Install dependencies in SDK Manager

4. Android Publishing

1. Installation

In this chapter, we are going to walk through the process of downloading and installing all necessary dependencies for development. After following these instructions, we will have a signed application for Android devices, ready to be uploaded to Google Play.

1.1 Install Java

Download and install a recent release of the [Java SDK 7](#) for your computer.

Check **Accept License Agreement** and click the link of Windows x86 or x64, according to the version of your operating system.

Java SE Development Kit 7u75		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input checked="" type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux x86	119.43 MB	jdk-7u75-linux-i586.rpm
Linux x86	136.77 MB	jdk-7u75-linux-i586.tar.gz
Linux x64	120.83 MB	jdk-7u75-linux-x64.rpm
Linux x64	135.66 MB	jdk-7u75-linux-x64.tar.gz
Mac OS X x64	185.86 MB	jdk-7u75-macosx-x64.dmg
Solaris x86 (SVR4 package)	139.55 MB	jdk-7u75-solaris-i586.tar.Z
Solaris x86	95.87 MB	jdk-7u75-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.66 MB	jdk-7u75-solaris-x64.tar.Z
Solaris x64	16.38 MB	jdk-7u75-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	138.66 MB	jdk-7u75-solaris-sparc.tar.Z
Solaris SPARC	98.56 MB	jdk-7u75-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	jdk-7u75-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.37 MB	jdk-7u75-solaris-sparcv9.tar.gz
Windows x86	127.8 MB	jdk-7u75-windows-i586.exe
Windows x64	129.52 MB	jdk-7u75-windows-x64.exe

Image 1: Download Java SDK.

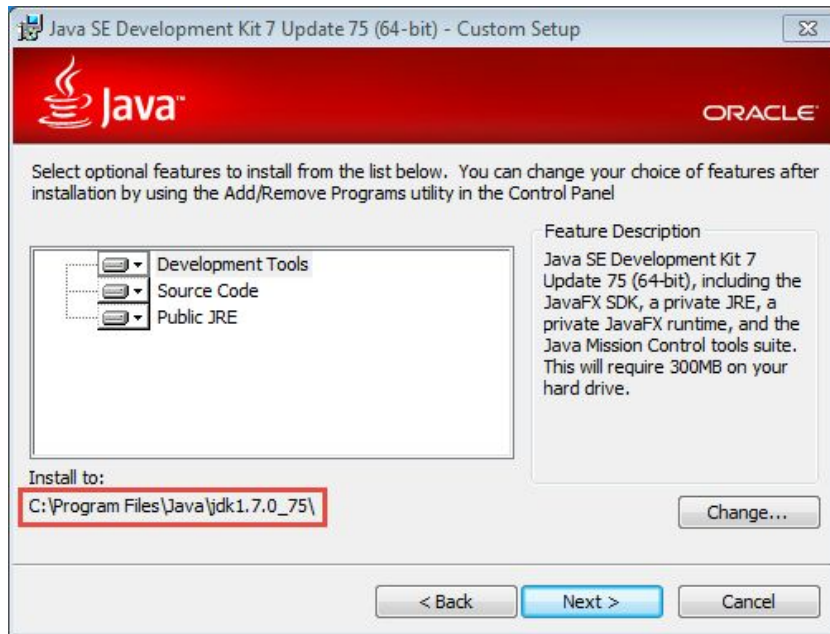


Image 2: JDK default installation path

1.2 Install the Android SDK

Download the [Android SDK](#).

Platform	Package	Size	SHA-1 Checksum
Windows	installer_r24.1.2-windows.exe (Recommended)	111364285 bytes	e0ec864efa0e7449db2d7ed069c03b1f4d36f0cd
	android-sdk_r24.1.2-windows.zip	159778618 bytes	704f6c874373b98e061fe2e7eb34f9fcb907a341
Mac OS X	android-sdk_r24.1.2-macosx.zip	89151287 bytes	00e43ff1557e8cba7da53e4f64f3a34498048256
Linux	android-sdk_r24.1.2-linux.tar.gz	168121600 bytes	68080e4e26e0e0182e1b1022e1b6b172e1240e51

Image 3: Download Android SDK

Before starting the installation, you must add a new user variable environment (See [section 1.7](#)) that points to the installation folder of Java SDK. Otherwise, the Android SDK installer throw an error saying that the **Java SDK was not found**.

Add as follow:

- Variable name: JAVA_HOME
- Variable value: C:\Program Files\Java\jdk1.7.0_XX (Replace XX with the version installed)

Run the Android SDK installer. In the installation dialog, choose **Install just for me** and change the install location to `C:\Android\android-sdk`.

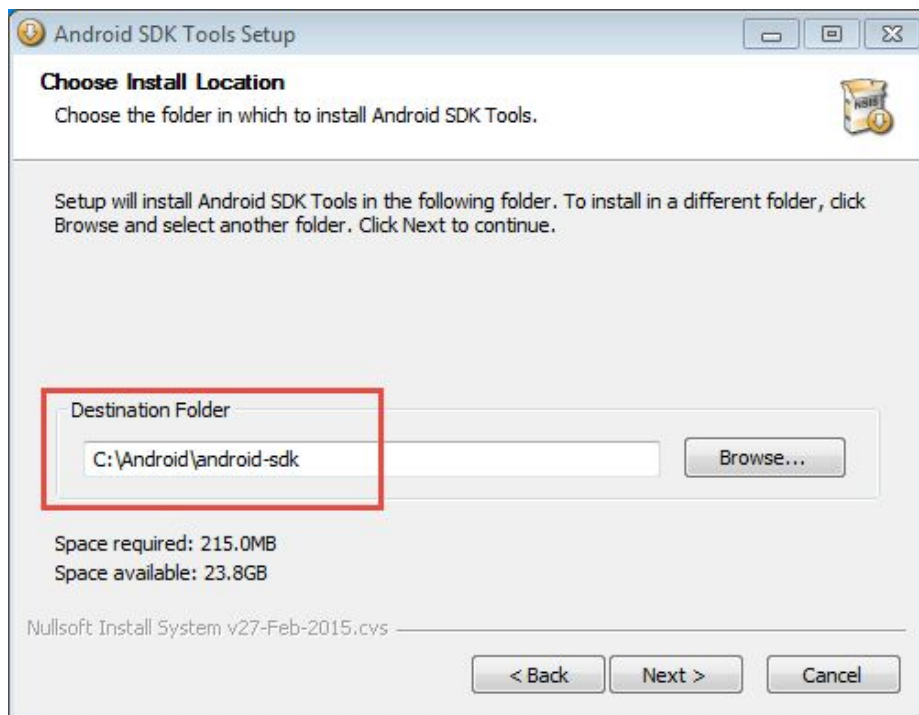


Image 4: Android SDK installation path

At the end of the process, uncheck the **Start SDK Manager** option and press **Finish**.

1.3 Install Git

Download [Git for Windows](#). In the installation process, select **Use Git from the Windows Command Prompt** (See image 5) .

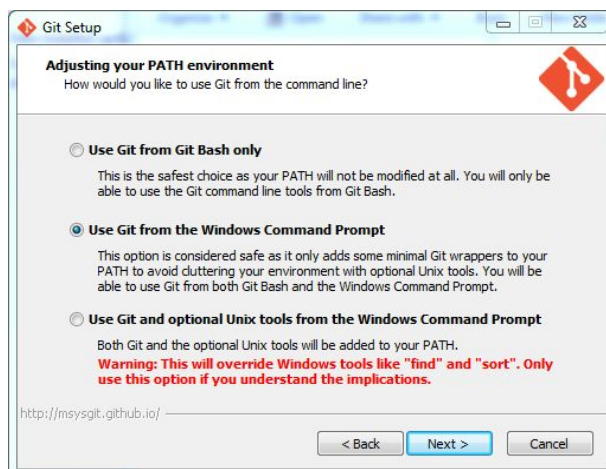


Image 5: Git path environment configuration

1.4 Install Node.js

Download [Node.js](#).

The installation folder must be *C:\Program Files\nodejs*

1.5 Install Dependencies

Open a new cmd.exe window, and run:

```
npm install -g yo gulp bower  
npm install -g generator-m-ionic
```

Go to nearme project folder and run:

```
npm install  
bower install
```

1.6 Environment variables

Now next part in adding and checking **Environment Variables** for all above locations.

Press Windows + R, type **control sysdm.cpl,,3** in the input dialog and press **Ok**.

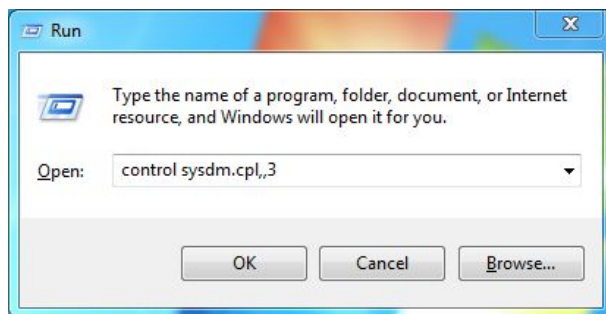


Image 6: Open System Properties from Run dialog

Press **Environment variables** in **System Properties** window.

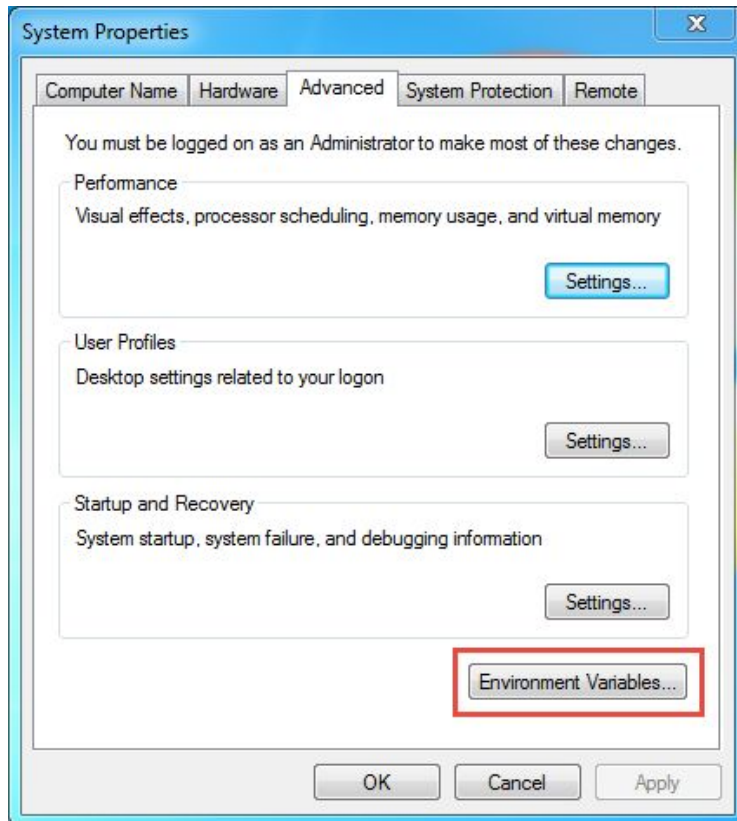


Image 7: Open Environment Variables dialog

In user variables, click on **New** and add as follow:

Variable name: ANDROID_HOME

Variable value: C:\Android\android-sdk

Then select **Path** variable and press **Edit**.

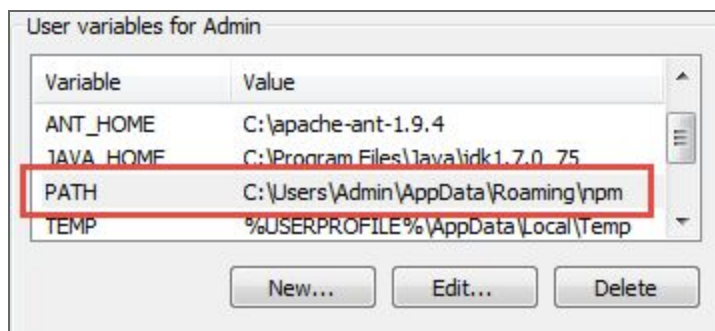


Image 8: Edit PATH variable

Add following values:

```
; %ANDROID_HOME%\platforms;%ANDROID_HOME%\platform-tools;%ANDROID_HOME%\tools;%ANDROID_HOME%\build-tools\23.0.1;%JAVA_HOME%\bin
```

Press **Ok** to save changes.

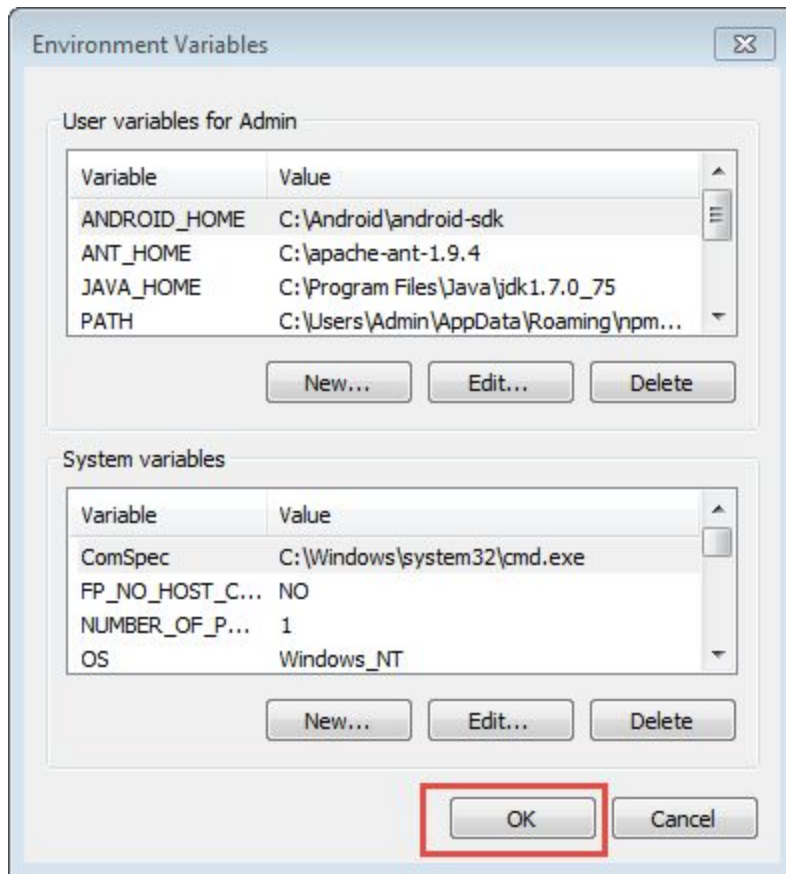


Image 9: Save environment variables

2. Configuration

2.1 App configuration

Open `env-dev.json` in `app/main/constants`. In this file you can setup the following:

- Parse: App ID and Server URL (required)
- Pushwoosh: App ID and Google Project Number (optional)
- Admob: Interstitial ID (iOS/Android) and Banner ID (optional)
- Google Analytics: Mobile Tracking ID (optional)
- Theme: balanced, calm, positive or any Ionic color. (Default: nearme)
- Unit: mi or km
- mapType: normal or satellite
- statusBarColor: Any HEX color

Open `config.xml` file and edit with your app information.

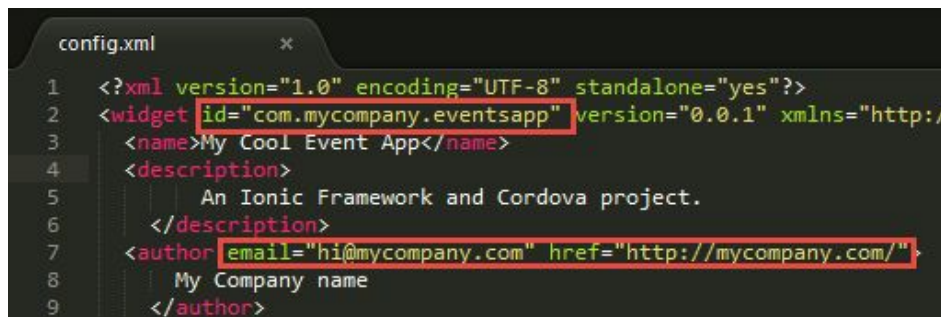


Image 10: Editing `config.xml` file in Sublime Text

Also, you need to enter your Facebook App Name/App ID and your Google Maps API keys.



2.2 Theme

The easiest way to change the main color of the app is editing the `nearme.scss` file and change the value of `$nearme` and `$nearme-dark` variables with your desired colors.

```
1 //nearme Theme
2 // Colors
3 // -----
4 $nearme: #e91e63 !default;
5 $nearme-dark: darken($nearme, 30%);
6
```

2.3 Google Analytics

Log in or create an account in [Google Analytics](#).

In the control panel, press the **Admin** tab located in the top menu. Then, press the drop-down of the **Account** section and click on **Create new account** option.

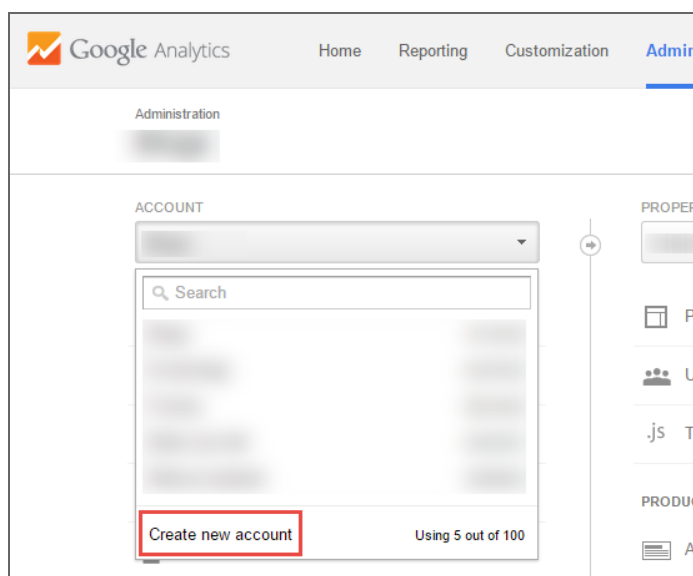


Image 11: Create new account option from Account section

In the new window, select **Mobile app**, and enter your account and app name. Fill the other fields and press **Get Tracking ID**.

Google Analytics Home Reporting Customization Admin

Administration > New Account

New Account

What would you like to track?

Website **Mobile app**

Setting up your account

Account Name required
Accounts are the top-most level of organization and contain one or more tracking IDs.

MyEvent

Setting up your property

App Name

MyEventsApp

Image 12: New Account form

A new window is open with your **Tracking ID**.

Administration

MyEvent / MyEventsApp

PROPERTY

MyEventsApp

Tracking ID

UA-61663037-1

Mobile App tracking - Dov

Property Settings

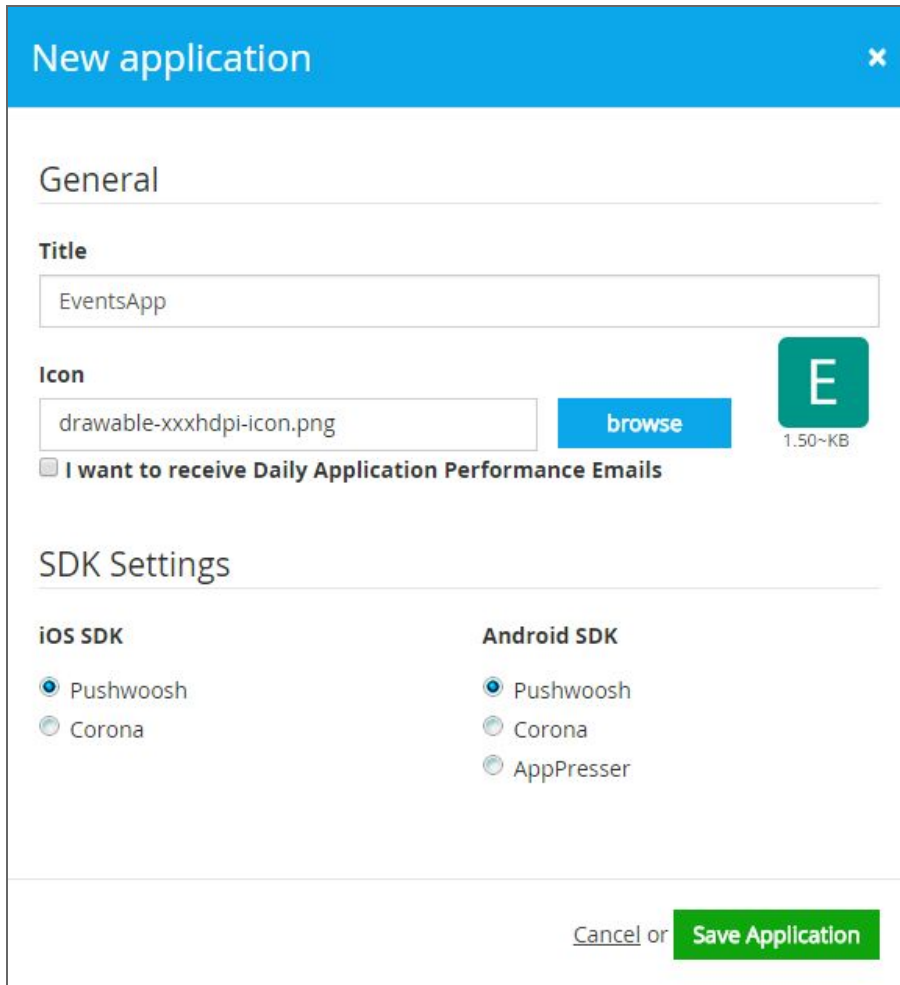
Here are the SDKs available for your an

Image 13: Tracking ID

2.4 Push Notifications

2.4.1 Configure PushWoosh

Create an account in [PushWoosh](#). Log in the administration panel and add a new app.



New application

General

Title

EventsApp

Icon

drawable-xxxhdpi-icon.png

browse

☐ I want to receive Daily Application Performance Emails

SDK Settings

iOS SDK

☒ Pushwoosh

☐ Corona

Android SDK

☒ Pushwoosh

☐ Corona

☐ AppPresser

Cancel or Save Application

Image 14: New application form in Pushwoosh

Enter the title, upload your app icon and select Pushwoosh in SDK Settings. Press **Save Application**.

Locate the **Application Code**.

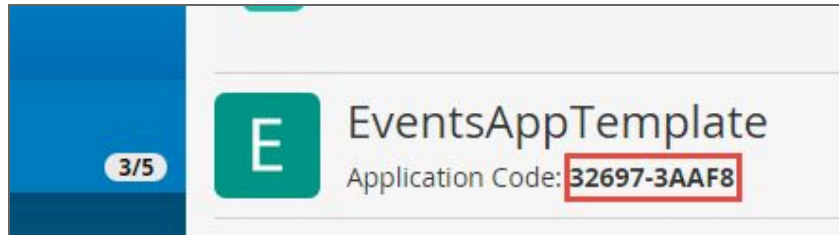


Image 15: Application Code in Pushwoosh

2.4.2 Configure Push notifications for Android

Go to [Developer Console](#) and create a new project. The project name can be whatever you want, ideally the name of your app.

A screenshot of the 'New Project' form in the Google Developer Console. The form has a title 'New Project'. Below it is a label 'Project name' with a help icon. A text input field contains the text 'EventsApp'. Below the input field, it says 'Your project ID will be eventsapp-907' followed by a help icon and an 'Edit' link. Below this is a link 'Show advanced options...'. At the bottom are two buttons: 'Create' (blue) and 'Cancel' (white with a grey border).

Image 16: New project form in Google Developer Console

Next, locate your Google Project Number. The Project Number is automatically assigned by the Google Developers Console when you create a project. You can find the Project Number in the **Overview** tab of the Google API console.

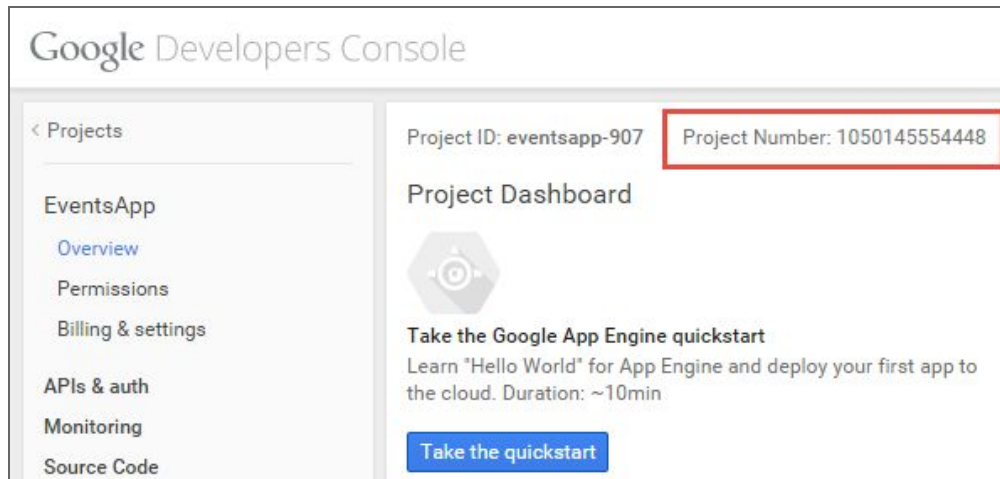


Image 17: Project Number in Google Developer Console

In the main Google APIs Console page, click **APIs & auth**, and select **APIs**. Locate **Google Cloud Messaging for Android** and click on it.

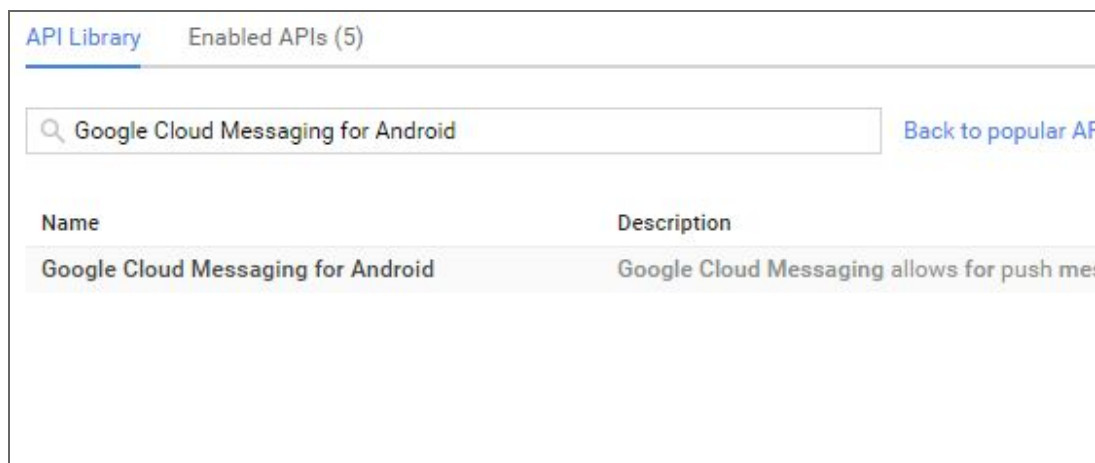


Image 18: Search Google Cloud Messaging for Android in API Library

Press **Enable API**.

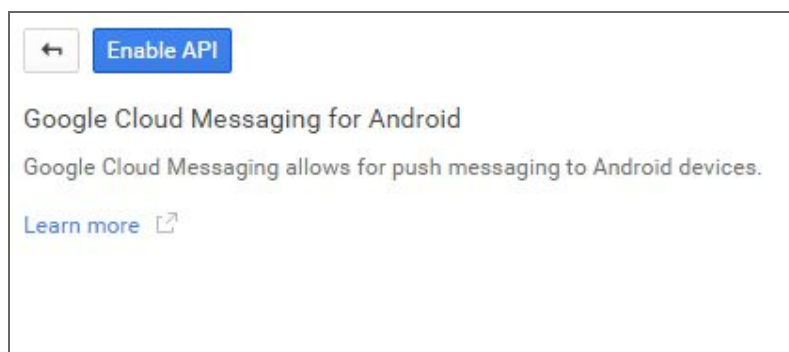


Image 19: Enable Google Cloud Messaging for Android

Now you need to create the Server Key. Go to the **Credentials** section and press **Create new key**.

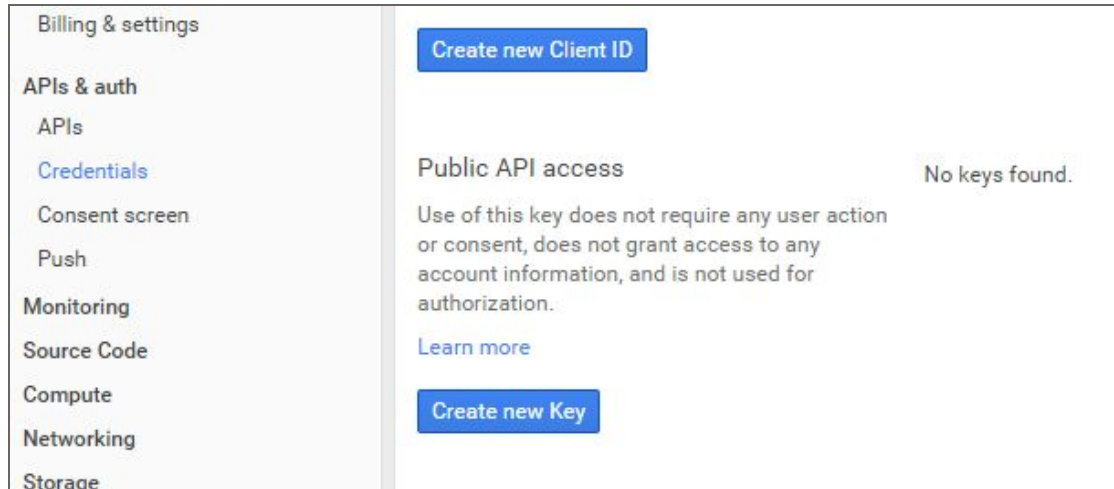


Image 20: Create new Key in Credentials section

Either a server key or a browser key should work. The advantage to using a server key is that it allows you to whitelist IP addresses. For now, select server key and press **Create**.

Here's the **API Key** you will need to configure your application in Pushwoosh Control Panel.

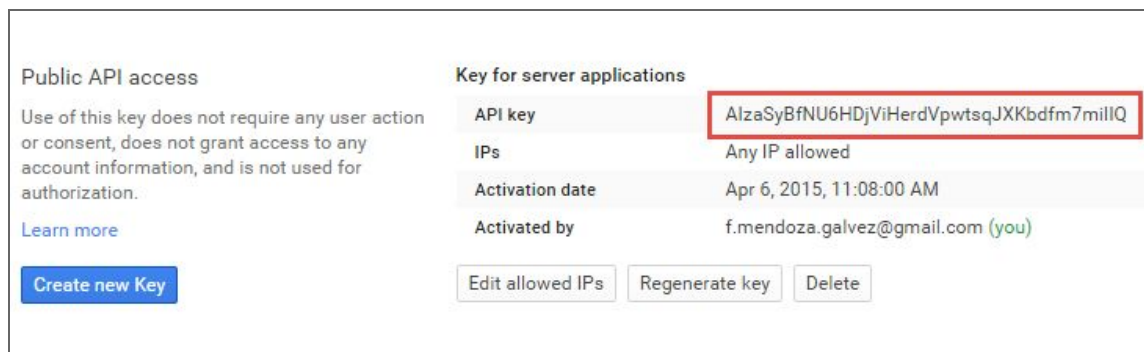
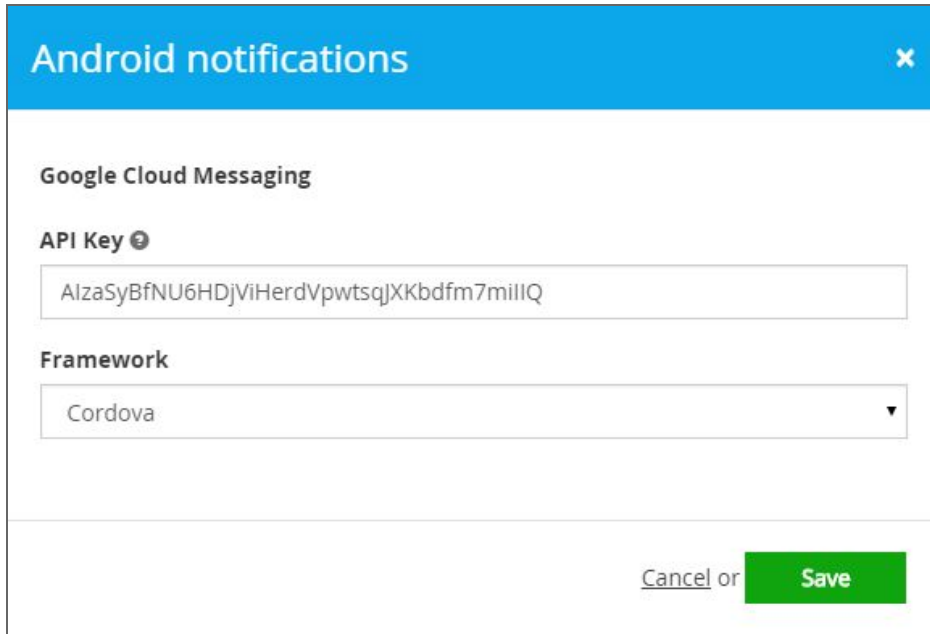


Image 21: API key of Public API access

Go to the Pushwoosh Control Panel. In your application click on **Android->Edit** to change the configuration for Android application.



Android notifications ✕

Google Cloud Messaging

API Key ⓘ

AlzaSyBfNU6HDjViHerdVpwtSqjXKbdfm7mIIQ

Framework

Cordova ▼

Cancel or **Save**

Image 22: Set API key in Pushwoosh Control Panel

Copy your API Key to the field provided and press **Save**.

2.5 AdMob

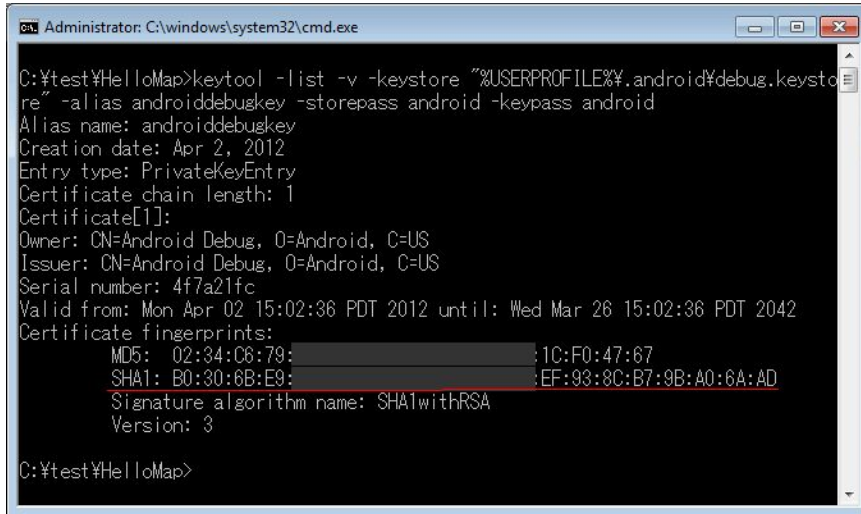
You have to create a new ad unit id for Android in [AdMob](#). Then, open `env-dev.json` and add your ad unit id.

2.6 Google Maps

2.6.1 Print certificate fingerprint

Open CMD and run:

```
keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

```
C:\test\HelloMap>keytool -list -v -keystore "%USERPROFILE%\android\debug.keystore" -alias androiddebugkey -storepass android -keypass android
Alias name: androiddebugkey
Creation date: Apr 2, 2012
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: 4f7a21fc
Valid from: Mon Apr 02 15:02:36 PDT 2012 until: Wed Mar 26 15:02:36 PDT 2042
Certificate fingerprints:
    MD5: 02:34:C8:79: [REDACTED] : 1C:F0:47:67
    SHA1: B0:30:6B:E9: [REDACTED] : EF:93:8C:B7:9B:A0:6A:AD
    Signature algorithm name: SHA1withRSA
    Version: 3

C:\test\HelloMap>
```

Take note of SHA1 fingerprint.

2.6.2 Obtain Google Maps API Key for Android

Go to [Google Developer Console](#) and create a new project.

Turn on **Google Maps Android API V2**.

Go to Credentials section and create a new Android key. Enter your SHA1 fingerprint and the package name of your app.

Take note of the API key.

Finally, open *config.xml* and look for *plugin.google.maps*. Replace `API_KEY_FOR_ANDROID` with yours.

Before submitting your app to Google Play, remember to repeat the 2.6.1 step with your release key and submit the SHA1 fingerprint to Google Developer Console.

3. Android Building

3.1 Install dependencies in SDK Manager

Open cmd.exe and type *android* to open the SDK Manager. Select the latest **Android SDK Build-tools**, **Android SDK Platform-tools** and **SDK platform** (API 23).

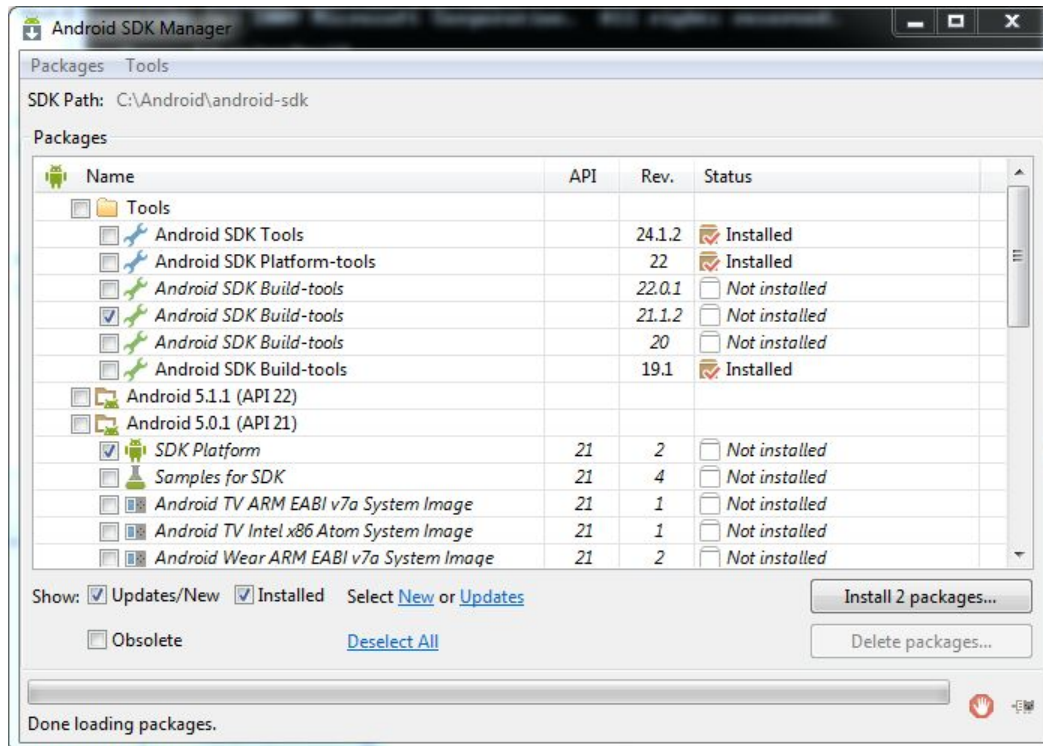


Image 23: Install Android SDK Build-tools and SDK Platform

It's also important to install the **Android Support Repository**, **Android Support Library**, **Google Play Services** and **Google Repository**.

Now open cmd.exe in **Nearme** folder and run:

```
gulp --cordova "prepare"
```

Now it's time to deploy in your device or emulator:

```
gulp --cordova "run android"
```

4. Android Publishing

To generate a release build for Android, we can use the following cordova cli command:

```
gulp --cordova "build android --release"
```

This will generate two releases. One for devices with the arm CPU architecture and one for devices with the x86 architecture.

We can find *android-armv7-release-unsigned.apk* and *android-x86-release-unsigned.apk* in *platforms/android/build/outputs/apk*.

Now, we need to sign the unsigned APKs and run an alignment utility on it to optimize it and prepare it for Google Play. If you already have a signing key, skip these steps and use that one instead.

Let's generate our private key using the *keytool* command that comes with the JDK. Open Terminal in nearme folder and run:

```
keytool -genkey -v -keystore my-release-key.keystore -alias nearmeappkey -keyalg RSA  
-keysize 2048 -validity 10000
```

You'll first be prompted to create a password for the keystore. Then, answer the rest of the nice tool's questions and when it's all done, you should have a file called *my-release-key.keystore* created in the current directory.

Note: Make sure to save this file somewhere safe, if you lose it you won't be able to submit updates to your app!

To sign both APKs, in the same Terminal app window, run the *jarsigner* tool which is also included in the JDK:

For ARM:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore  
my-release-key.keystore  
platforms/android/build/outputs/apk/android-armv7-release-unsigned.apk nearmeappkey
```

For x86:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore  
my-release-key.keystore  
platforms/android/build/outputs/apk/android-x86-release-unsigned.apk nearmeappkey
```

This signs the apks in place. Finally, we need to run the *zipalign* tool to optimize the APK:

For ARM:

```
zipalign -v 4 platforms/android/build/outputs/apk/android-armv7-release-unsigned.apk  
NearmeApp-armv7.apk
```

For x86:

```
zipalign -v 4 platforms/android/build/outputs/apk/android-x86-release-unsigned.apk  
NearmeApp-x86.apk
```

Now we have our final release binaries called *NearmeApp-armv7.apk* and *NearmeApp-x86* and we can release this on the Google Play.

To start, you'll need to visit the [Google Play Store Developer Console](#) and create a new developer account. Unfortunately, this is not free. However, the cost is only \$25 compared to Apple's \$99.

Once you have a developer account, you can go ahead and click **Publish an Android App on Google Play**.

Then, you can go ahead and click the button to edit the store listing (we will upload an APK later). You'll want to fill out the description for the app.

When you are ready, upload your APK and publish the listing.