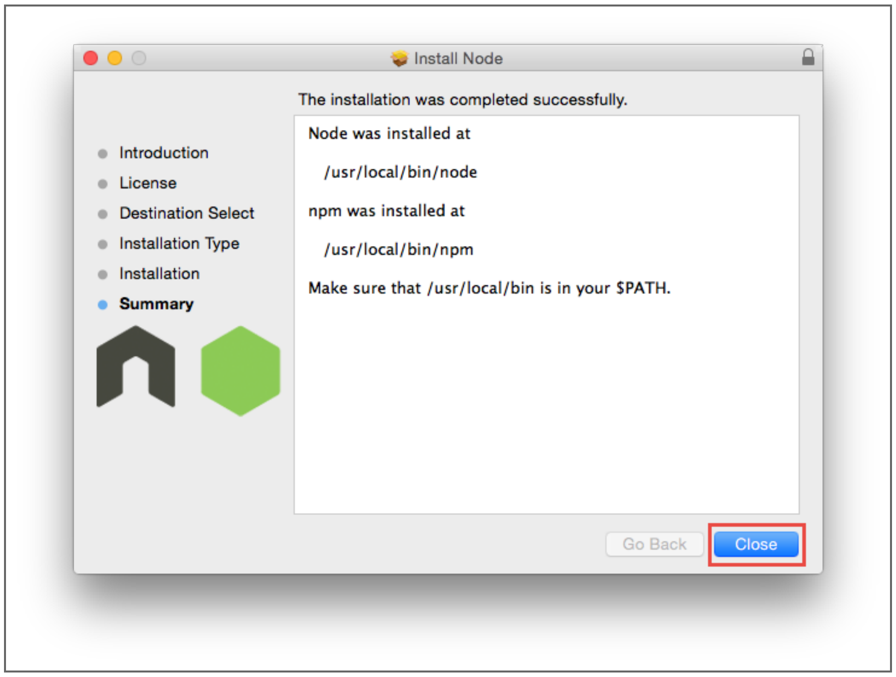
Nearme App
Documentation for Mac OS X

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 Apple and Android devices, ready to be uploaded to App Store and Google Play, respectively.

1.1 Install Node.js

Download Node.js. Double click the pkg file to open the installer window.



The installation was completed successfully.

Node was installed at

/usr/local/bin/node

npm was installed at

/usr/local/bin/npm

Make sure that /usr/local/bin is in your $PATH.

Image 1: Node.js installer

1.2 Install Cordova and Ionic

We are going to install Generator M Ionic to create, build and deploy the app. Open Terminal and run:

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

If you get permissions errors, try to run the commands with *sudo*. Go to nearme folder and run:

```
npm install
bower install
```

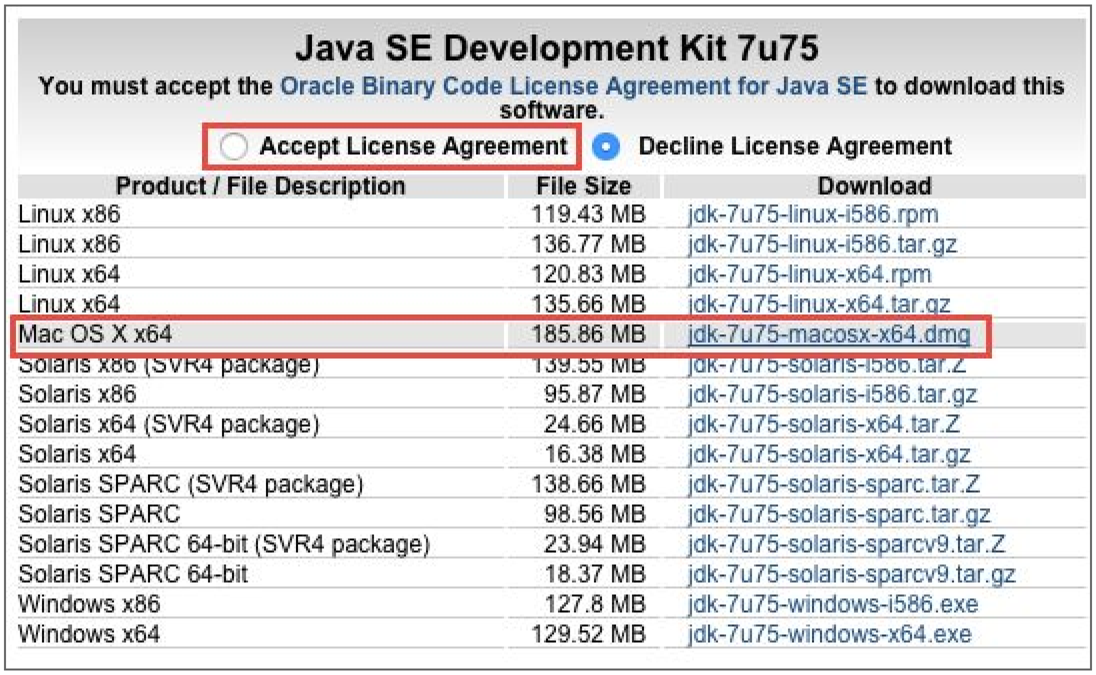
This will install the necessary dependencies to run the app.

1.3 Android dependencies

1.3.1 Install Java

Download and install a recent release of the Java SDK 7 for your computer.

Check **Accept License Agreement** and download the binaries for Mac OS X.



Java SE Development Kit 7u75

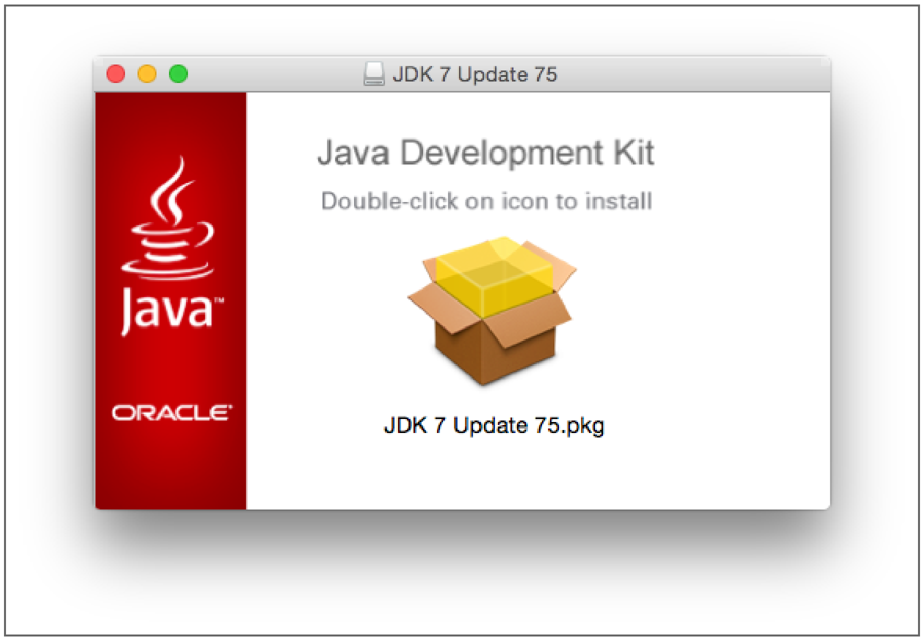
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ 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 2: Download Java SDK

Open DMG file to begin the installation.

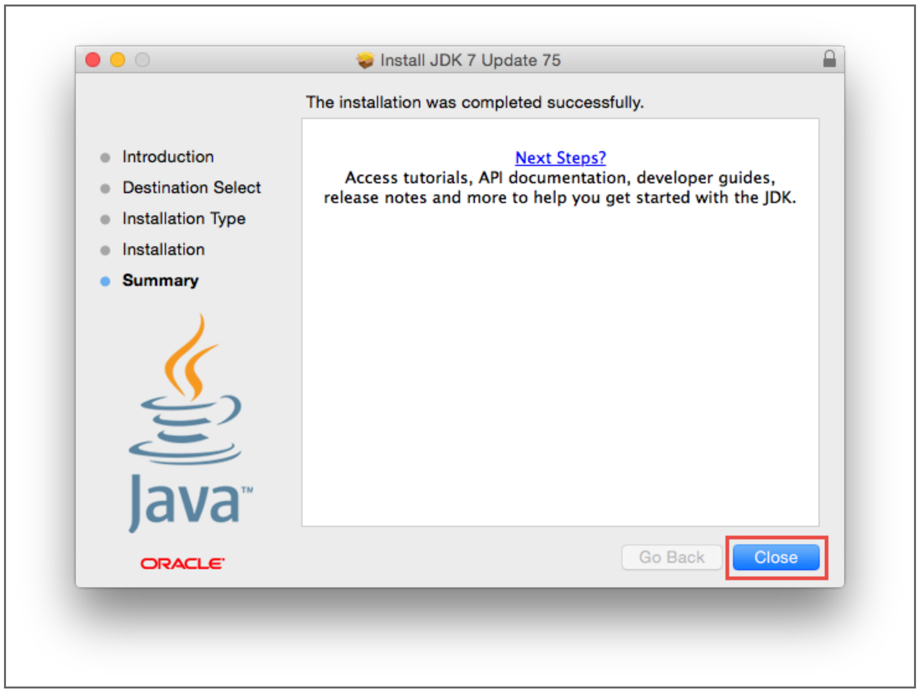


Java Development Kit

Double-click on icon to install

JDK 7 Update 75.pkg

Image 3: JDK installer



The installation was completed successfully.

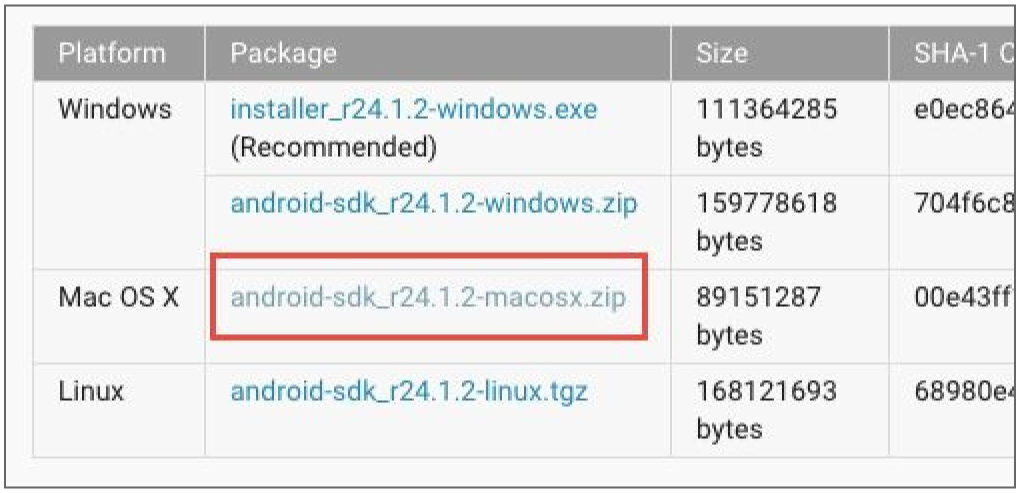
Next Steps?

Access tutorials, API documentation, developer guides, release notes and more to help you get started with the JDK.

Image 4: JDK installer

1.3.2 Install the Android SDK

Download the Android SDK for Mac OS X.



Platform	Package	Size	SHA-1 C
Windows	installer_r24.1.2-windows.exe (Recommended)	111364285 bytes	e0ec864
	android-sdk_r24.1.2-windows.zip	159778618 bytes	704f6c8
Mac OS X	android-sdk_r24.1.2-macosx.zip	89151287 bytes	00e43ff
Linux	android-sdk_r24.1.2-linux.tgz	168121693 bytes	68980e

Image 5: Download Android SDK

Extract the zip file and copy the extracted folder into *Dev f*older. The folder structure should be as follows:

```
/Users/user/Dev/android-sdk
```

Open *~/.bash_profile* and add:

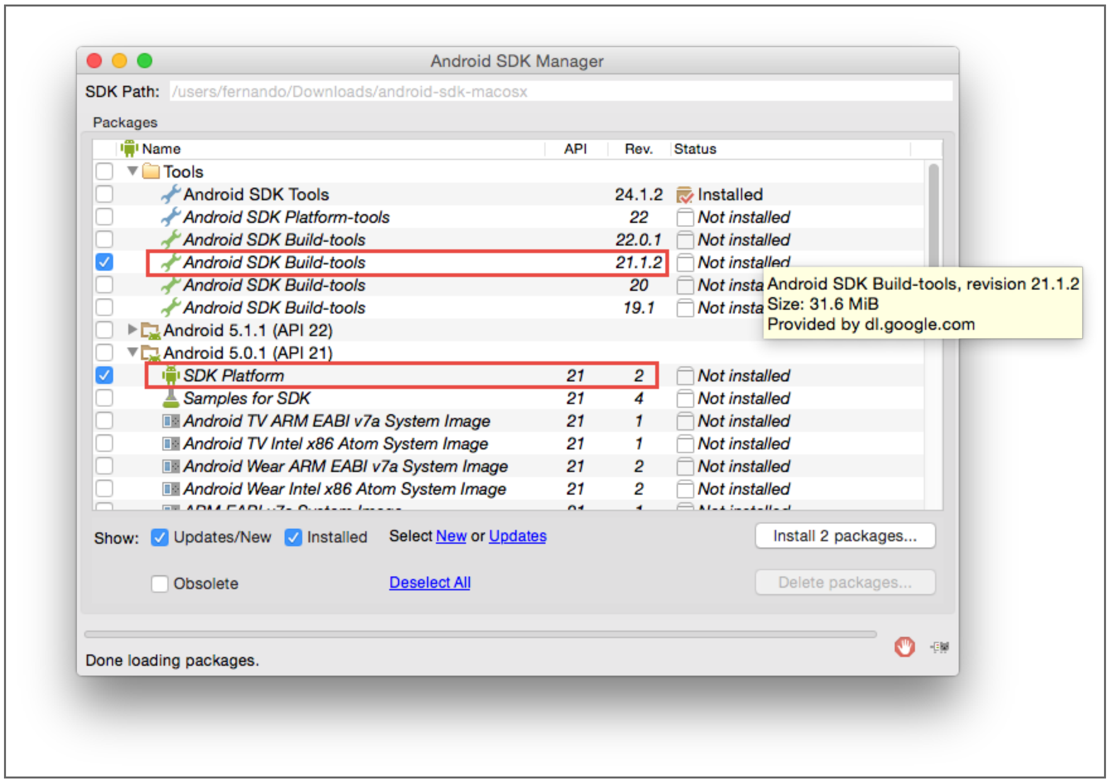
```
export
PATH=${PATH}:/users/user/Dev/android-sdk/tools:/users/user/Dev/android-sdk/platform-tools
```

In Terminal, type the following to save changes:

```
source ~/.bash_profile
```

1.3.3 Install Android dependencies

Open Terminal and type `android` to start the SDK Manager. Select the latest **Android SDK Build-tools, Android SDK Platform-tools** and **SDK platform** (API 23).



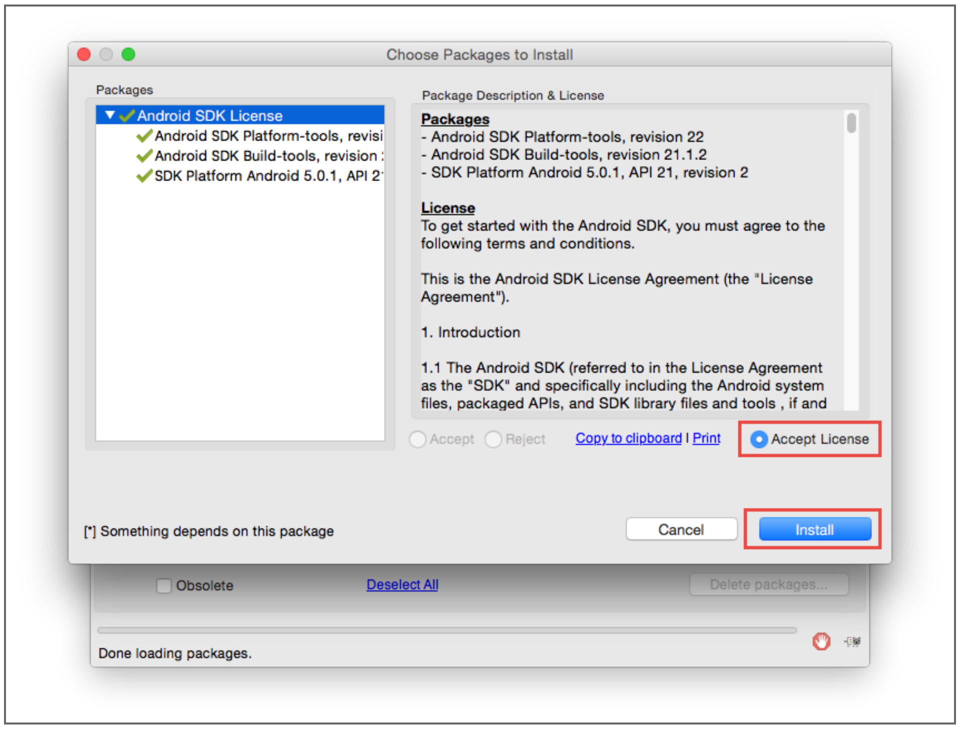
	Name	API	Rev.	Status
☐	Tools			
☐	Android SDK Tools		24.1.2	Installed
☐	Android SDK Platform-tools		22	Not installed
☐	Android SDK Build-tools		22.0.1	Not installed
✓	Android SDK Build-tools		21.1.2	Not installed
☐	Android SDK Build-tools		20	Not insta
☐	Android SDK Build-tools		19.1	Not insta
☐	Android 5.1.1 (API 22)			
☐	Android 5.0.1 (API 21)			
✓	SDK Platform	21	2	Not installed
☐	Samples for SDK	21	4	Not installed
☐	Android TV ARM EABI v7a System Image	21	1	Not installed
☐	Android TV Intel x86 Atom System Image	21	1	Not installed
☐	Android Wear ARM EABI v7a System Image	21	2	Not installed
☐	Android Wear Intel x86 Atom System Image	21	2	Not installed

Android SDK Build-tools, revision 21.1.2
Size: 31.6 MiB
Provided by dl.google.com

Image 6: Install Android-23

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

Press **Install packages**. In the new window, accept license and press **Install**.


Choose Packages to Install
Packages
Android SDK License
Android SDK Platform-tools, revisi
Android SDK Build-tools, revision
SDK Platform Android 5.0.1, API 2
Package Description & License
Accept
Reject
Copy to clipboard | Print
Accept License
[*] Something depends on this package
Cancel
Install
Obsolete
Deselect All
Delete packages...
Done loading packages.

Packages
- Android SDK Platform-tools, revision 22
- Android SDK Build-tools, revision 21.1.2
- SDK Platform Android 5.0.1, API 21, revision 2

License
To get started with the Android SDK, you must agree to the following terms and conditions.

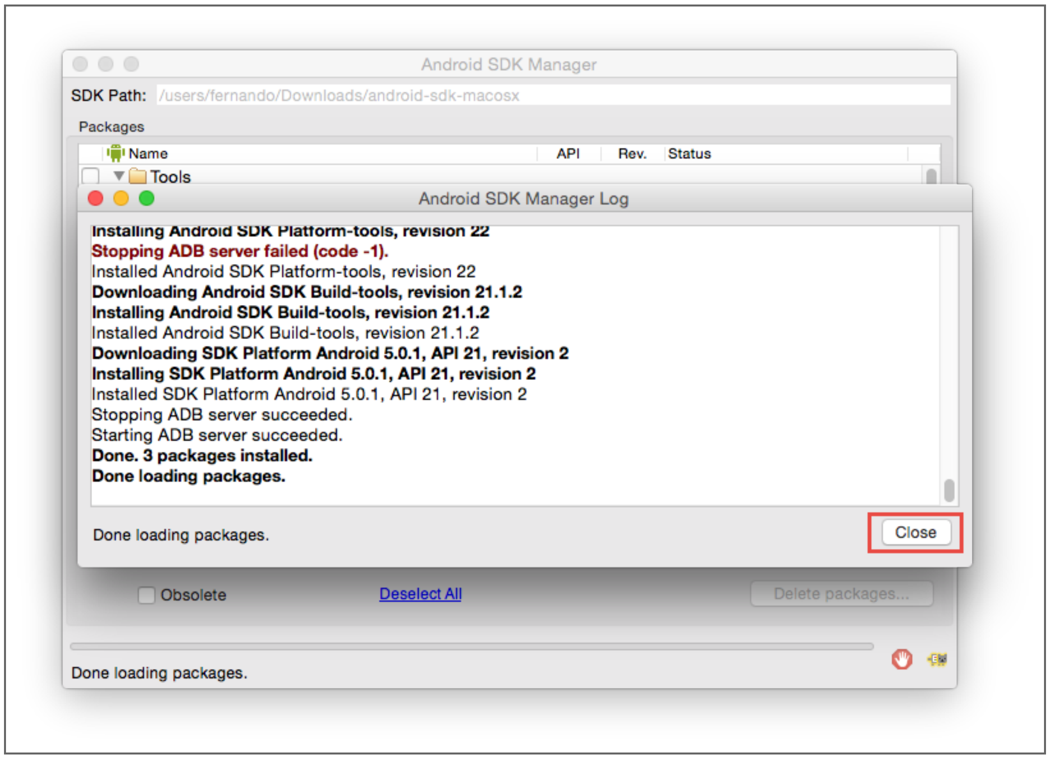
This is the Android SDK License Agreement (the "License Agreement").

1. Introduction

1.1 The Android SDK (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and SDK library files and tools , if and

Image 7: Install Android-23

Finally, press **Close**.



```
Installing Android SDK Platform-tools, revision 22
Stopping ADB server failed (code -1).
Installed Android SDK Platform-tools, revision 22
Downloading Android SDK Build-tools, revision 21.1.2
Installing Android SDK Build-tools, revision 21.1.2
Installed Android SDK Build-tools, revision 21.1.2
Downloading SDK Platform Android 5.0.1, API 21, revision 2
Installing SDK Platform Android 5.0.1, API 21, revision 2
Installed SDK Platform Android 5.0.1, API 21, revision 2
Stopping ADB server succeeded.
Starting ADB server succeeded.
Done. 3 packages installed.
Done loading packages.
```

Done loading packages.

Image 8: Close SDK Manager

1.4 iOS Dependencies

1.4.1 Install Xcode

Install XCode from App Store. This will take a while since it is ~2 gigs in size.

Image 9: Install Xcode

Once install is completed, open Xcode and accept the license.

1.4.2 Install IOS-SIM

Install **ios-sim** from npm. We need this to launch the IOS simulator from command line. Open Terminal app and execute the following:

```
npm install -g ios-sim
```

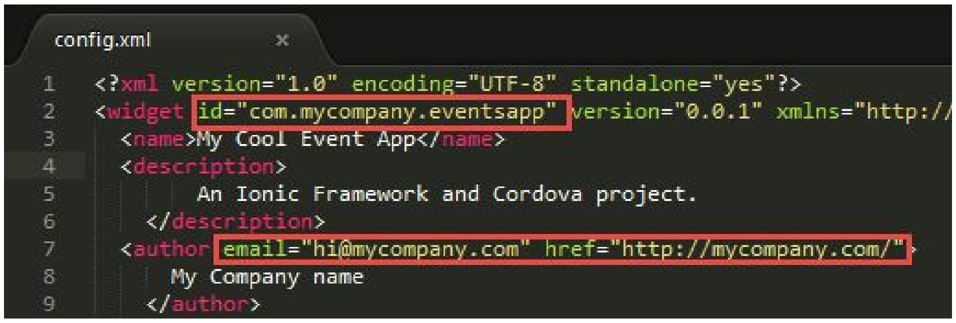
2. Configuration

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

Now open config.xml file and edit with your app information.

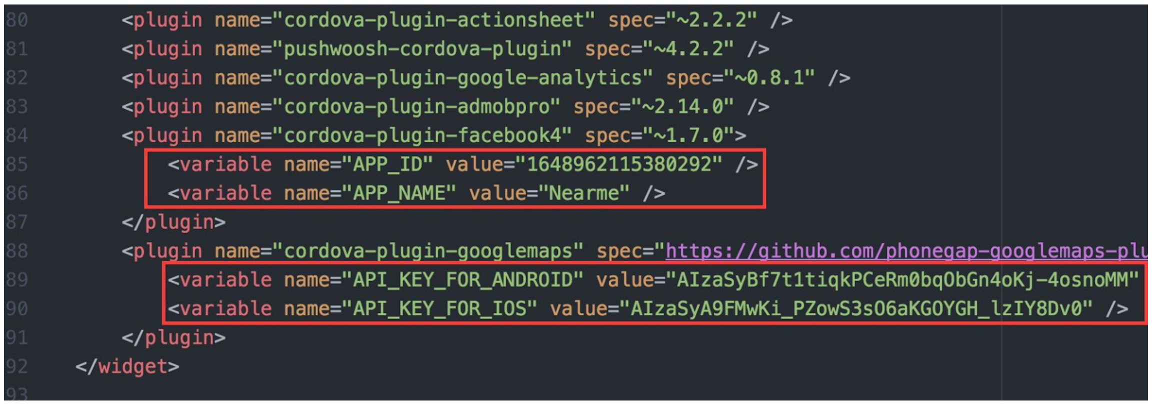


config.xml ×

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<widget id="com.mycompany.eventsapp" version="0.0.1" xmlns="http://
  <name>My Cool Event App</name>
  <description>
      An Ionic Framework and Cordova project.
    </description>
  <author email="hi@mycompany.com" href="http://mycompany.com/">
      My Company name
    </author>
```

Image 10: Config.xml file

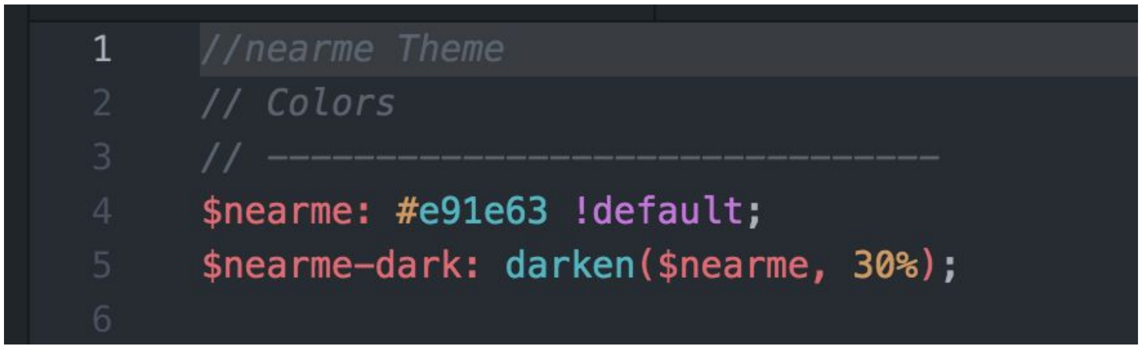
Also, you need to enter your Facebook App Name/App ID and your Google Maps API keys.



```xml
        <plugin name="cordova-plugin-actionsheet" spec="~2.2.2" />
        <plugin name="pushwoosh-cordova-plugin" spec="~4.2.2" />
        <plugin name="cordova-plugin-google-analytics" spec="~0.8.1" />
        <plugin name="cordova-plugin-admobpro" spec="~2.14.0" />
        <plugin name="cordova-plugin-facebook4" spec="~1.7.0">
            <variable name="APP_ID" value="1648962115380292" />
            <variable name="APP_NAME" value="Nearme" />
        </plugin>
        <plugin name="cordova-plugin-googlemaps" spec="https://github.com/phonegap-googlemaps-plu
            <variable name="API_KEY_FOR_ANDROID" value="AIzaSyBf7t1tiqkPCeRm0bq0bGn4oKj-4osnoMM"
            <variable name="API_KEY_FOR_IOS" value="AIzaSyA9FMwKi_PZowS3s06aKGOYGH_lzIY8Dv0" />
        </plugin>
    </widget>
```

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.



```scss
//nearme Theme
// Colors
// ---------------------------------
$nearme: #e91e63 !default;
$nearme-dark: darken($nearme, 30%);
```

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. Press the drop-down of the **Account** section and click the **Create new account** option.

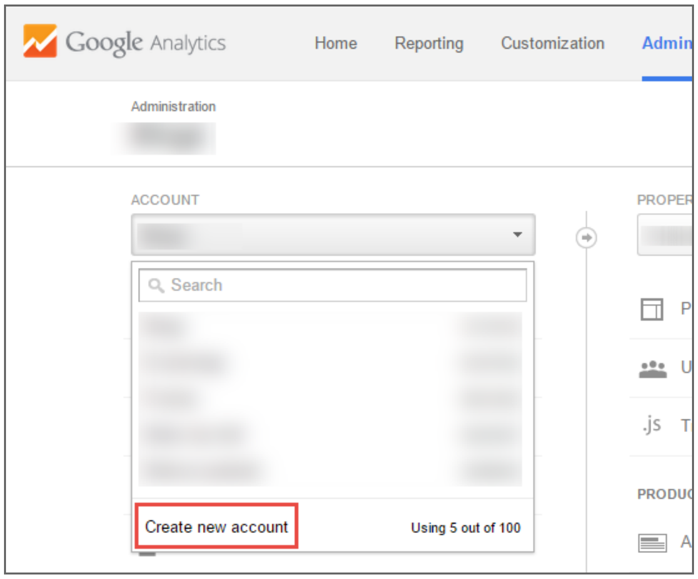
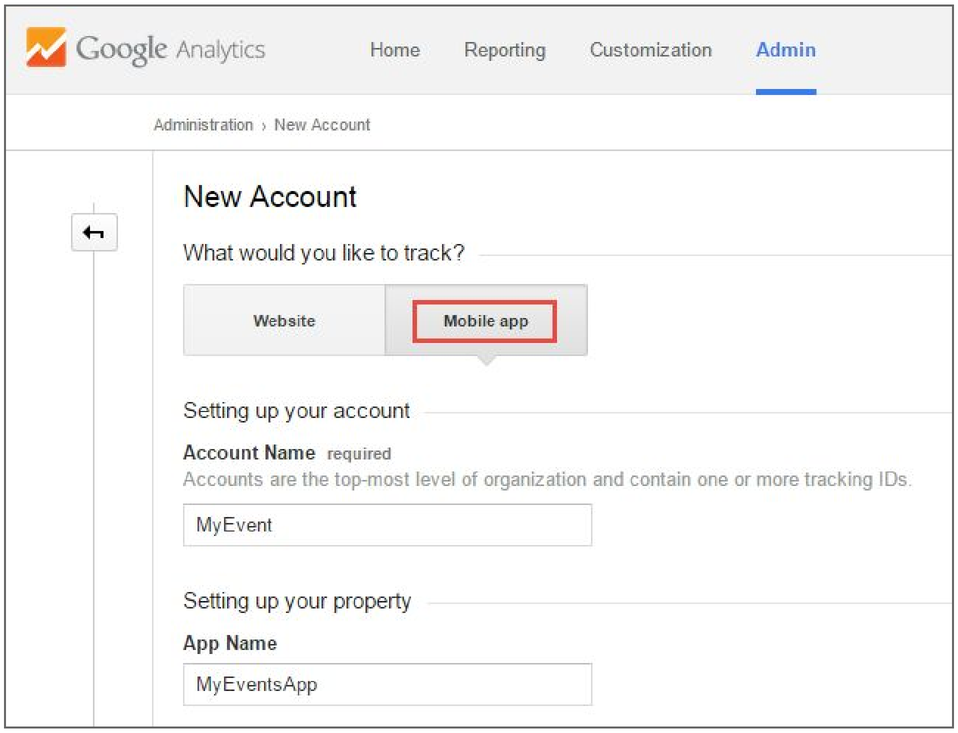


Image 11: Create new account option from Account section

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



New Account

What would you like to track?

Setting up your account

Account Name required

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

MyEvent

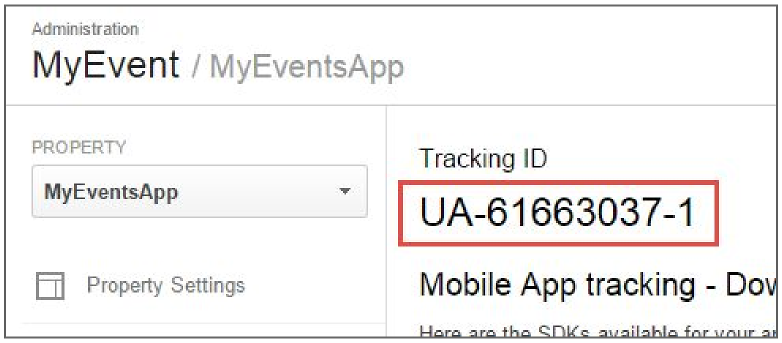
Setting up your property

App Name

MyEventsApp

Image 12: New Account form

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



MyEvent / MyEventsApp

PROPERTY

MyEventsApp

Tracking ID

UA-61663037-1

Mobile App tracking - Dov

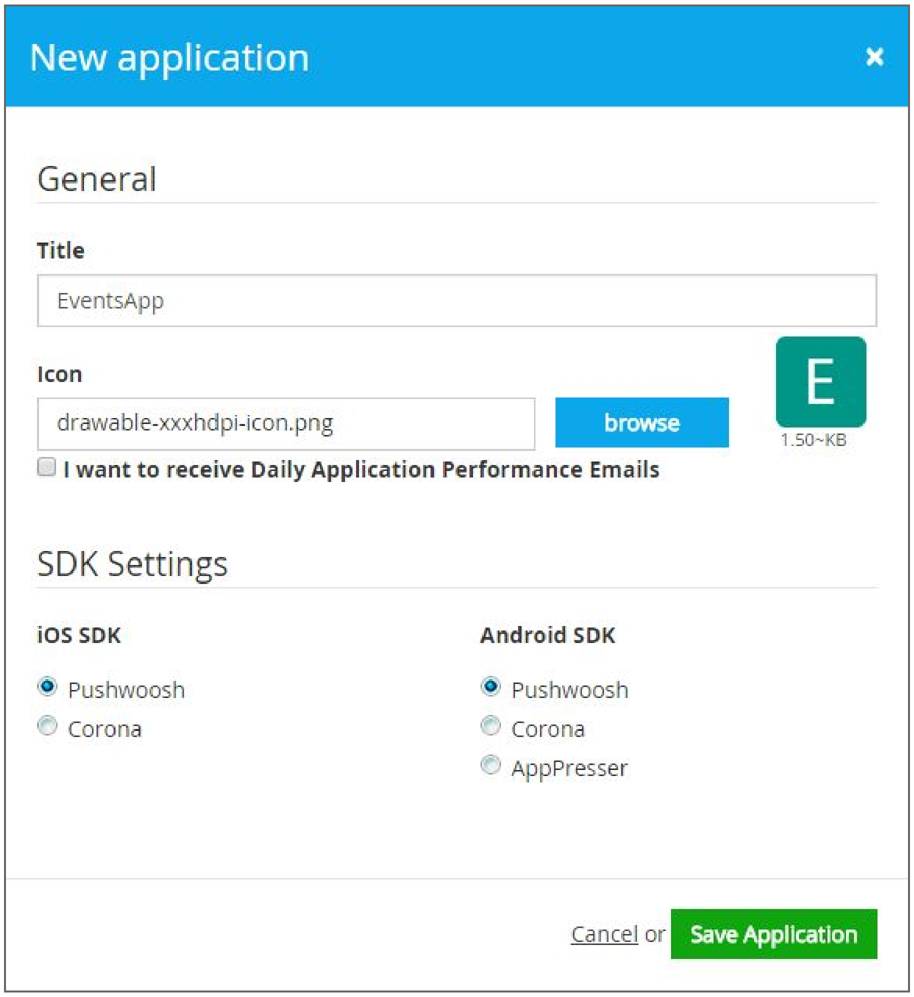
Here are the SDKs available for your a

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

1.50~KB

I want to receive Daily Application Performance Emails

SDK Settings

iOS SDK: [✓ Pushwoosh] Corona

Android SDK: [✓ Pushwoosh] Corona AppPresser

Image 14: New application form in Pushwoosh

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

Take note of the **Application Code**.

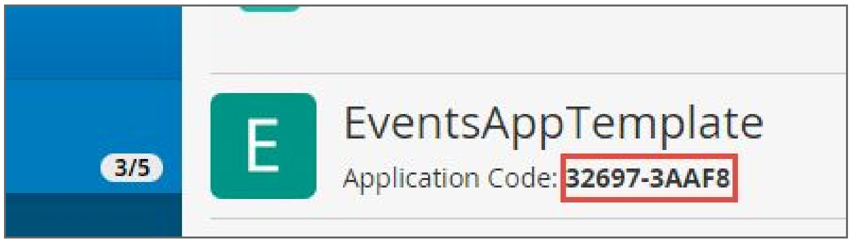


Image 15: Application Code in Pushwoosh

2.4.2 Configure Push notifications for Android

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

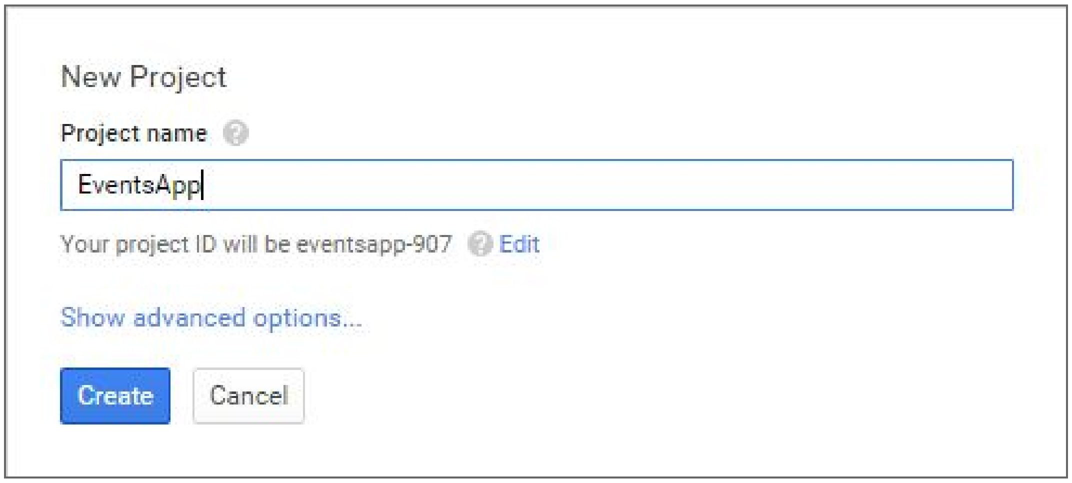
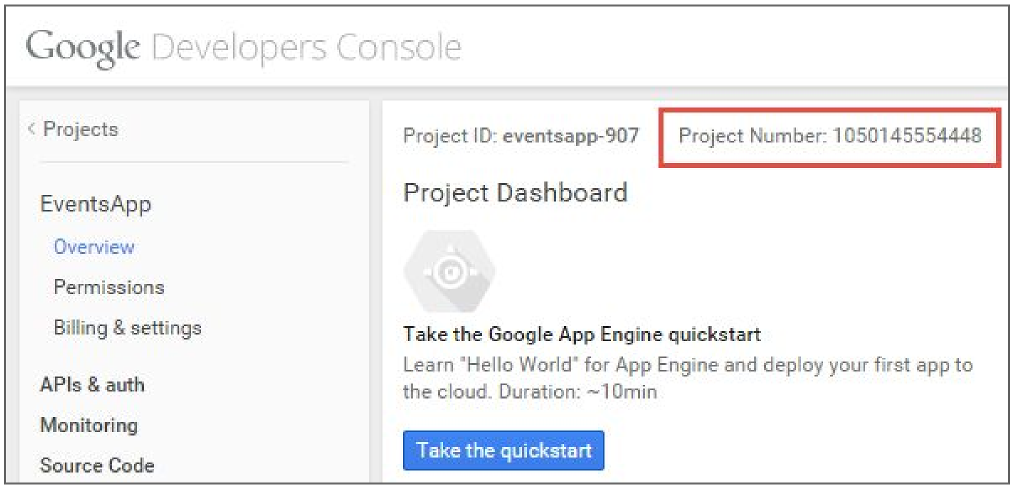


Image 16: New project form in Google Developer Console

Next, take note of your Google Project Number, you will need it for push notifications. 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.



Project ID: eventsapp-907 Project Number: 1050145554448

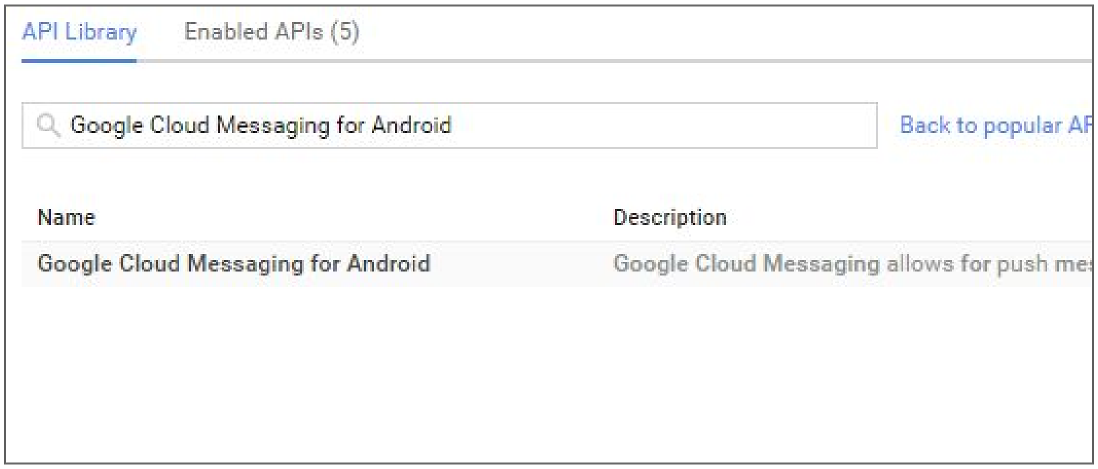
Project Dashboard

Take the Google App Engine quickstart

Learn "Hello World" for App Engine and deploy your first app to the cloud. Duration: ~10min

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.

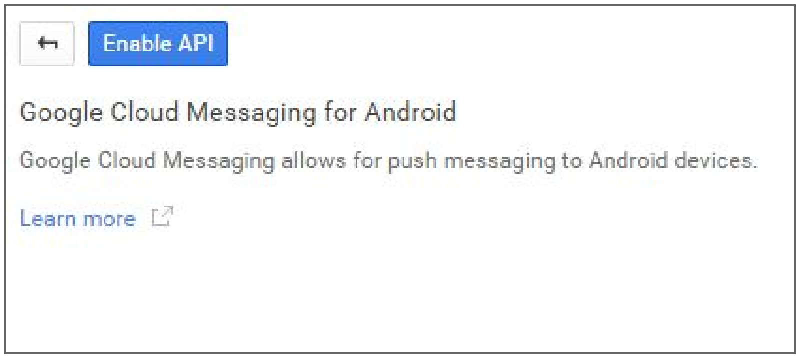


Google Cloud Messaging for Android

Name	Description
Google Cloud Messaging for Android	Google Cloud Messaging allows for push mes

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

Press **Enable API**.

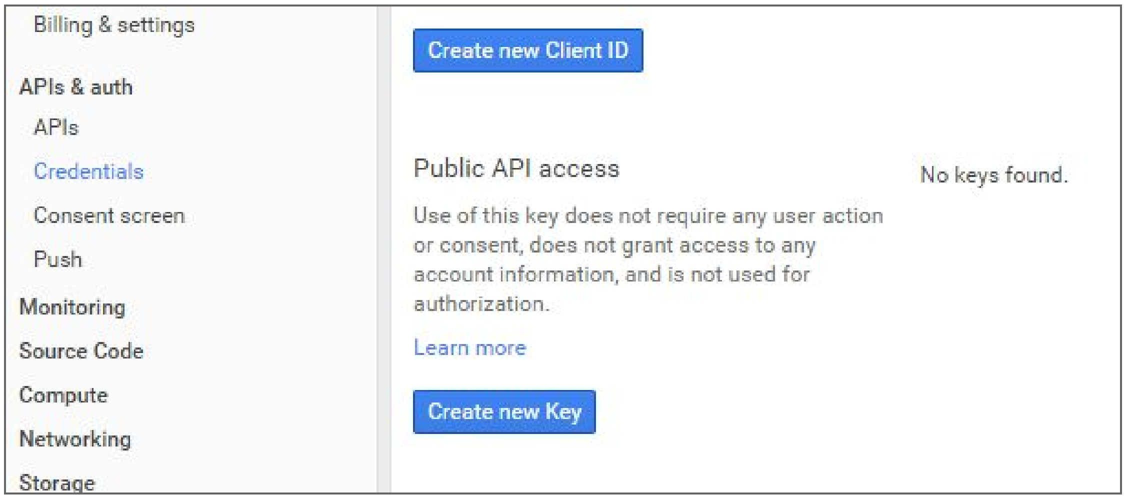


Google Cloud Messaging for Android

Google Cloud Messaging allows for push messaging to Android devices.

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**.

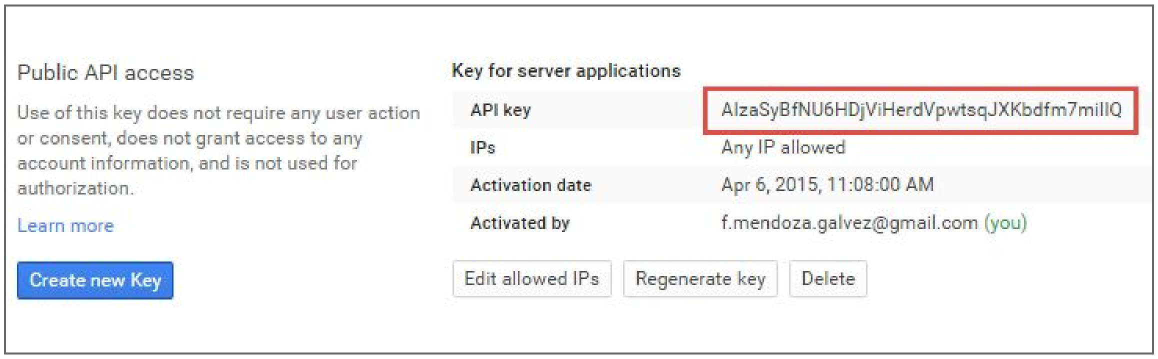


Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

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.



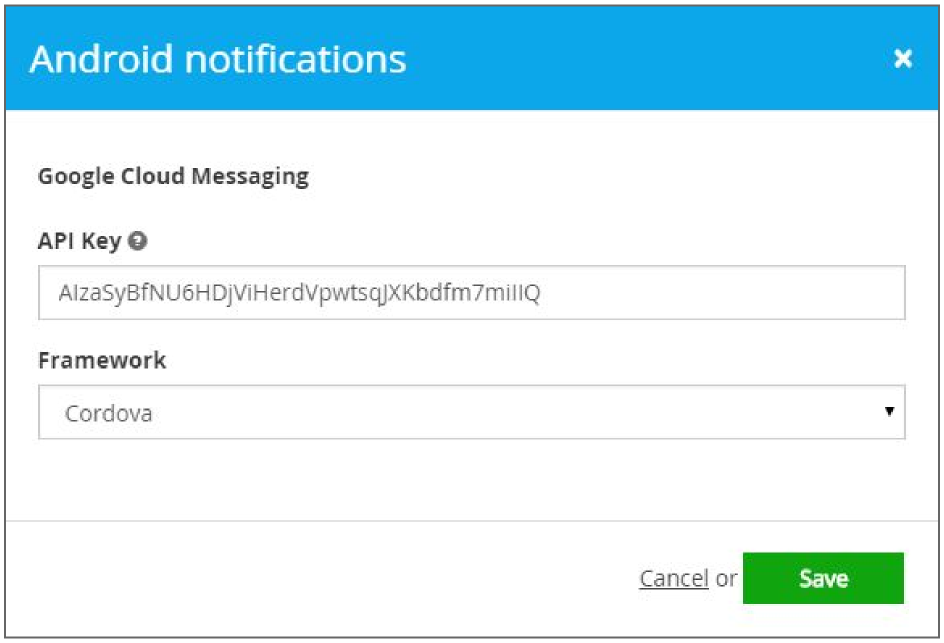
Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.

Key for server applications

API key	AIzaSyBfNU6HDjViHerdVpwtsqJXKbdfm7millQ
IPs	Any IP allowed
Activation date	Apr 6, 2015, 11:08:00 AM
Activated by	f.mendoza.galvez@gmail.com (you)

Image 21: API key of Public API access

Go to Pushwoosh Control Panel, select your app and click on **Android - Edit**.


Android notifications

Google Cloud Messaging

API Key

AIzaSyBfNU6HDjViHerdVpwtsqJXKbdfm7milIQ

Framework

Cordova

Image 22: Save API key in Pushwoosh Control Panel

Enter your API Key and press **Save**.

2.4.3 Configure Push notifications for iOS

To configure Pushwoosh for your iOS application you will need the following:

- Apple Push Notification Service (APNS) certificate
- private key for the certificate
- password for the private key

The process might look a bit complicated but actually it is not that hard to do.

If you have a Premium account in Pushwoosh, you can skip to 2.4.3.6 step.

2.4.3.1 Generating a Certificate Request Link

Launch the Keychain Access application in your Mac OS X.

Select **Keychain Access**, **Certificate Assistant**, **Request a Certificate From a Certificate Authority**.

Enter the information and check the **Saved to disk** option. Click **Continue**.

Save the certificate request using the suggested name and click **Save**. Click **Done** in the next screen.

2.4.3.2 Creating an App ID Link

If you have App ID ready, just skip to step Configuring an App ID for Push Notifications

Each iOS applications that uses the APNs must have a unique application ID that uniquely identifies itself. In this step, you will learn how to create an App ID for push notifications.

Sign in to the iOS Developer Center.

Click on the **Certificates, Identifiers & Profiles.** Choose **Identifiers** from the iOS apps list.

Click on the **App IDs** tab on the left and then click on the **New App ID** button.

Enter your **App ID Description** (Ex: “NearmeAppID”), and select an **App ID Prefix**. In the **App ID Suffix** section choose **Explicit App ID**, and provide the correct Bundle Identifier in the form com.company.application.

Check **Push Notifications** from the list of **App Services** and click **Continue**, then **Submit**, and then **Done**.

You should now see the App ID that you have created.

2.4.3.3. Configuring an App ID for Push Notifications Link

Once an App ID is created, you need to configure it for push notifications.

To configure an App ID for push notification, you need to click the **Create Certificate**.

You will now see the **Apple Push Notification service SSL Certificate Assistant** screen. Click **Continue**.

Click the **Choose File** button to locate the Certificate Request file that you have saved earlier. Click **Generate**.

Choose the certificate file from your drive and click **Generate**.

Your SSL Certificate will now be generated. Click **Continue**, and then click the **Download** button to download the SSL Certificate on your drive. Click **Done**.

This is the first file your will need to upload to Pushwoosh.

Download the certificate and double-click on it to install it in the Keychain Access application. This is the SSL certificate that will be used by Pushwoosh so that it can contact the APNs to send push notifications to your applications. Now you need to export the certificate.

2.4.3.4. Exporting Private Key

Open up the Keychain Access Application and select the **Keys** category.

Click on the private key associated with your iOS Push Certificate and click **Export Items** in the menu.

Please note this is not a private key for your iOS Development certificate.

You will be prompted to create a password, you will need to enter the same password on Pushwoosh configuration page.

This is the second file you need to upload to Pushwoosh. You also need to enter the same password you have used to create the Private Key.

To install the Certificate and the Private Key for the application log into your Control Panel, go to **My applications**, select your application, and open the **Configure** page in the left menu.

2.4.3.5. Configure push notifications through the Control Panel

Select iOS platform by clicking the **Configure** button. Pushwoosh provides two ways to configure push notifications through the Control Panel: Manual and Automatic.

For Manual Configuration upload the certificate and the private key (remember those two files we have created few steps above?) from your computer. Make sure you select the correct gateway: Sandbox for Development certificates and developer app builds or Production (aka Distribution) gateway for AdHoc and AppStore builds, enter your **Private key password** and click **Save**.

If you want to switch gateway for Pushwoosh from Sandbox to Production (for example), check **Production** in the bottom and upload the correct Certificate and Private Key for the production environment.

For testing purposes we strongly recommend creating two versions of the app in Pushwoosh – one for production, and the other one for development with sandbox gateway and development certificates. Otherwise, there’s a big chance to lose live subscribers by accidentally invalidating device tokens because of sending a push to the wrong gateway.

2.4.3.6 Auto Configuration (available for Premium accounts)

For Auto Configuration select the **Auto** tab and follow the next steps:

1. Enter your Apple ID and Password
2. Select your Developer team
3. Select your App ID

4. Select the gateway you want to configure: Sandbox for developer app builds, Production (aka Distribution) gateway for AdHoc and AppStore builds
5. Make your Provisioning profile push compatible (do not use this step for refreshing the expired certificates)

2.5 AdMob

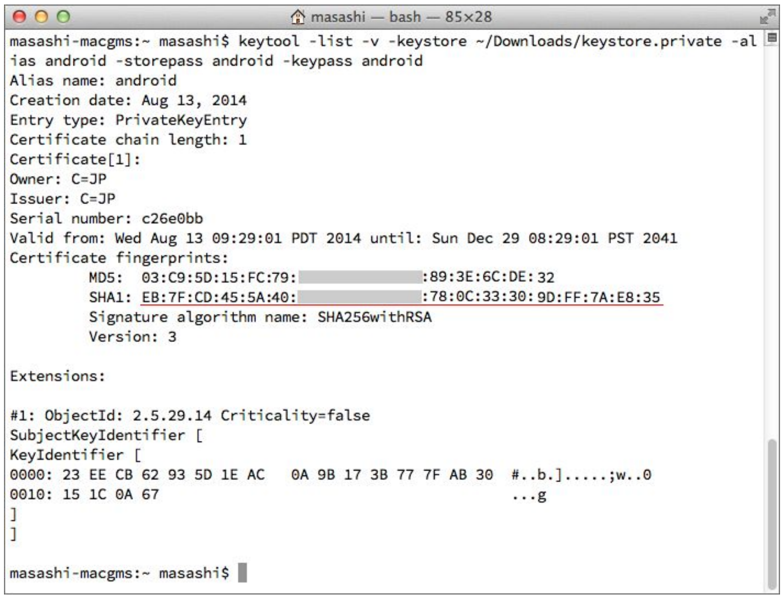
You have to create a new ad unit id in AdMob, one for Android and other for Apple devices. Then, open *env-dev.json* and add your ad unit id.

2.6 Google Maps

2.6.1 Print certificate fingerprint

Open Terminal and run:

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



```
masashi-macgms:~ masashi$ keytool -list -v -keystore ~/Downloads/keystore.private -al
ias android -storepass android -keypass android
Alias name: android
Creation date: Aug 13, 2014
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: C=JP
Issuer: C=JP
Serial number: c26e0bb
Valid from: Wed Aug 13 09:29:01 PDT 2014 until: Sun Dec 29 08:29:01 PST 2041
Certificate fingerprints:
         MD5:  03:C9:5D:15:FC:79:[redacted]:89:3E:6C:DE:32
         SHA1: EB:7F:CD:45:5A:40:[redacted]:78:0C:33:30:9D:FF:7A:E8:35
         Signature algorithm name: SHA256withRSA
         Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 23 EE CB 62 93 5D 1E AC   0A 9B 17 3B 77 7F AB 30  #..b.].....;w..0
0010: 15 1C 0A 67                                        ...g
]
]

masashi-macgms:~ masashi$ 
```

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.

2.6.3 Obtain Google Maps API Key for iOS

Turn on **Google Maps SDK for IOS**.

Go to Credentials section and create a new iOS key. Enter 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 and API_KEY_FOR_IOS with your API Keys.

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. Building

Open Terminal in nearme folder and run:

```
gulp --cordova ‘prepare’
```

This will create the android and iOS projects under *platforms* directory, and install the cordova plugins declared in config.xml file.

In Terminal, run the following command to build the app:

```
gulp --cordova ‘build android’
gulp --cordova ‘build ios’
```

Connect your device or open a new instance of a Genymotion emulator and run:

```
gulp --cordova ‘run android’
```

To launch the iOS simulator, run:

```
gulp --cordova ‘emulate ios’
```

4. Android Publishing

To generate a release build for Android:

```
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.

Then, 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.

5. iOS Publishing

Run the following in Terminal:

```
gulp --cordova ‘build ios --release’
```

This will generate an Xcode project inside of *platforms/ios* which you can open with Xcode and sign your app for publishing.