site/edit/master/content/docs/intro/migration/index.md)

Deploying to a Device

Testing your app in the browser with `ionic serve` or with an emulator is fast, easy and convenient when your app is in development, but eventually you're going to have to test on a device. Not only is it the only way to accurately test how your app will behave and perform, many Ionic Native (https://ionicframework.com/docs//native/) plugins will only work when they are run on actual hardware.

Android Devices

Deploying to an Android device is a fairly straightforward process. If you have a working Android development environment, you're ready to go.

Requirements

- Java JDK (http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html)
- Android Studio (https://developer.android.com/studio/index.html)
- Updated Android SDK tools, platform and component dependencies. Available through Android Studio's SDK Manager (https://developer.android.com/studio/intro/update.html)

Running Your App

To run your app, all you have to do is enable USB debugging and Developer Mode on your Android device, then run `ionic cordova run android --device` from the command line.

This will produce a debug build of your app, both in terms of Android and Ionic's code

Enabling USB debugging and Developer Mode can vary between devices, but is easy to look up with a Google search. You can also check out Enabling On-device Developer Options (https://developer.android.com/studio/run/device.html#developer-device-options) in the Android docs.

Production Builds

To run or build your app for production, run

```
ionic cordova run android --prod --release
# or
ionic cordova build android --prod --release
```

This will minify your app's code as Ionic's source and also remove any debugging capabilities from the APK. This is generally used when deploying an app to the Google Play Store.

Sign Android APK

If you want to release your app in the Google Play Store, you have to sign your APK file. To do this, you have to create a new certificate/keystore.

Let's generate your private key using the keytool command that comes with the JDK:

```
keytool -genkey -v -keystore my-release-key.jks -keyalg RSA -keysize 2048 -validit
```

You'll first be prompted to create a password for the keystore. Then, answer the rest of the nice tools's questions and when it's all done, you should have a file called my-release-key.jks 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 the unsigned APK, run the jarsigner tool which is also included in the JDK:

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore my-release-key.jk
```

This signs the APK in place. Finally, we need to run the zip align tool to optimize the APK. The zipalign tool can be found in `/path/to/Android/sdk/build-tools/VERSION/zipalign`. For example, on OS X with Android Studio installed, zipalign is in `~/Library/Android/sdk/build-tools/VERSION/zipalign`:

```
zipalign -v 4 app-release-unsigned.apk HelloWorld.apk
```

To verify that your apk is signed run apksigner. The apksigner can be also found in the same path as the zipalign tool:

```
apksigner verify HelloWorld.apk
```

Now we have our final release binary called HelloWorld.apk and we can release this on the Google Play Store for all the world to enjoy!

All steps can also be found here: Android SDK docs (https://developer.android.com/studio/publish/app-signing.html#signing-manually)

iOS Devices

Unlike Android, iOS developers need to generate a provisioning profile to code sign their apps for testing. The good news is that, as of iOS9, you can develop and test your apps on your iOS device without a paid Apple Developer account. This is particularly great for developers who want to try out mobile development with Ionic, since it saves the cost but still provides a lot of the features of having a full Apple Developer account. For a full breakdown of the features included, check out

Apple's docs

(https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/Sup

CH38-SW1).

Requirements

- Xcode 7 or higher
- iOS 9
- A free Apple ID (https://appleid.apple.com/) or paid Apple Developer account

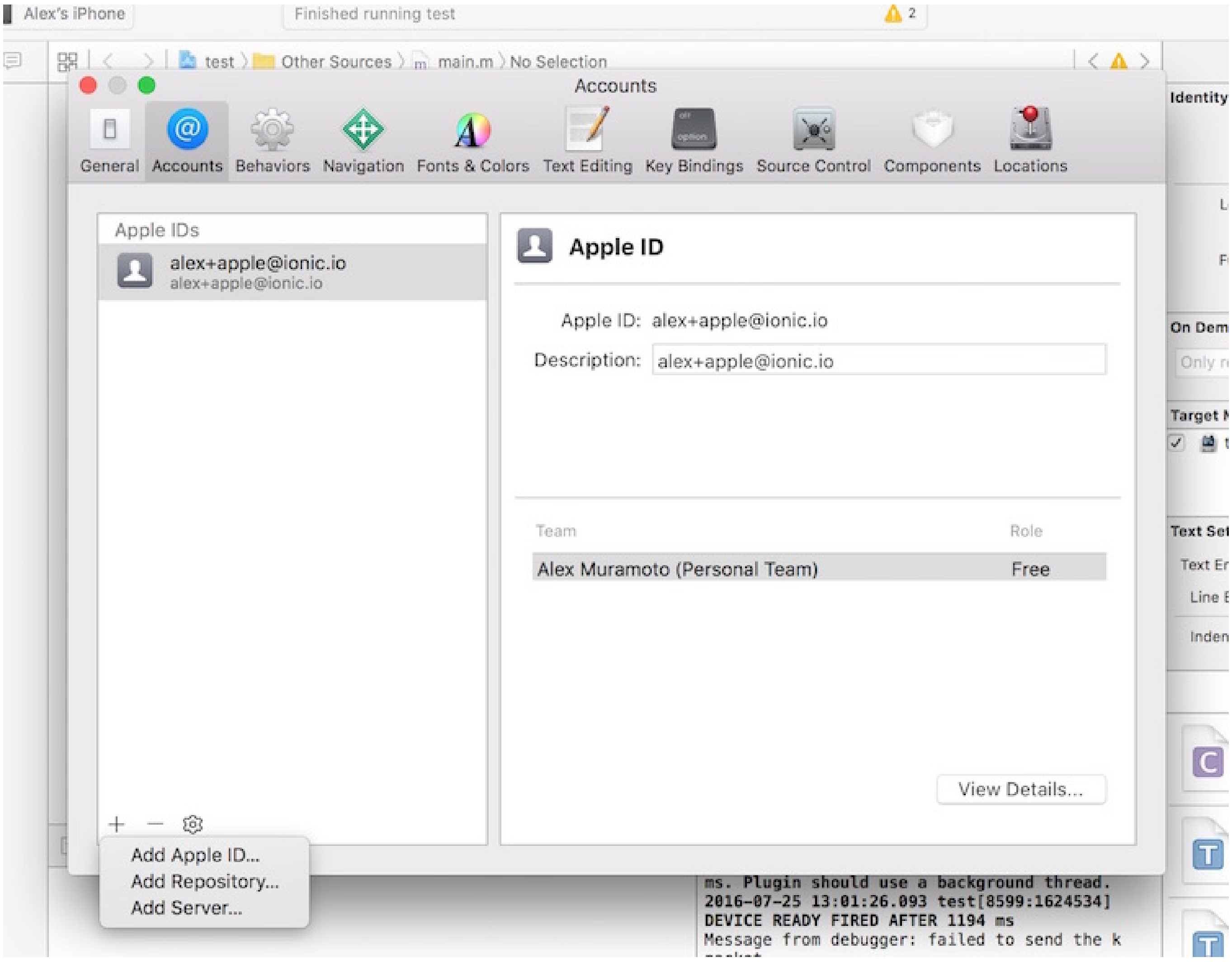
Creating a Provisioning Profile

To start, you'll need to set up a provisioning profile to code sign your apps.

Using an Apple ID

1. Open Xcode preferences (Xcode > Preferences…)
2. Click the 'Accounts' tab
3. Login with your Apple ID (+ > Add Apple ID…)

Once you've successfully logged in, a new 'Personal Team' with the role 'Free' will appear beneath your Apple ID.



Apple ID: alex+apple@ionic.io
Description: alex+apple@ionic.io

Team	Role
Alex Muramoto (Personal Team)	Free

```
ms. Plugin should use a background thread.
2016-07-25 13:01:26.093 test[8599:1624534]
DEVICE READY FIRED AFTER 1194 ms
Message from debugger: failed to send the k
```

Using an Apple Developer Account

Creating a provisioning profile with a paid Apple Developer account is a little bit more involved. For full instructions, check out Launching Your App on Devices (https://developer.apple.com/library/content/documentation/IDEs/Conceptual/AppDistributionGuide in the Apple Developer docs.

Running Your App

1. Run a production build of your app with `ionic cordova build ios --prod`
2. Open the `.xcodeproj` file in `platforms/ios/` in Xcode
3. Connect your phone via USB and select it as the run target
4. Click the play button in Xcode to try to run your app

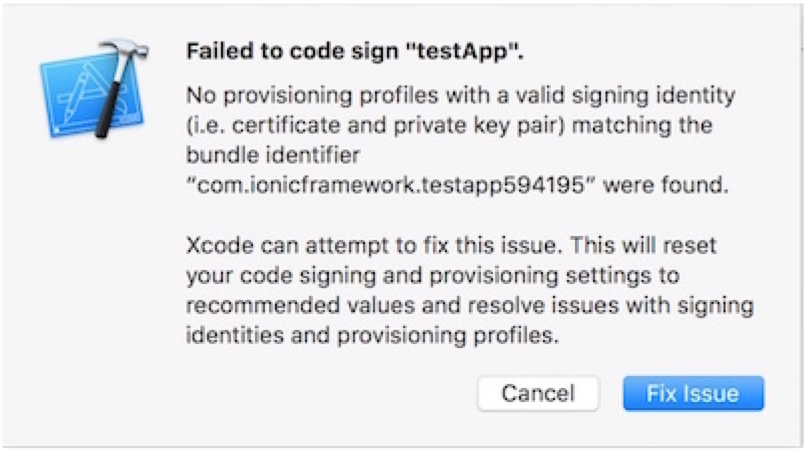
Oops, code signing error! No problem.

Code Signing Your App

Next, you'll need to code sign your app. How you do this will depend on if you are running Xcode 8 or an earlier version.

Xcode 7 and Earlier

If you are running Xcode 7 or earlier, you'll get a code signing error that looks like this when you try to run the app:

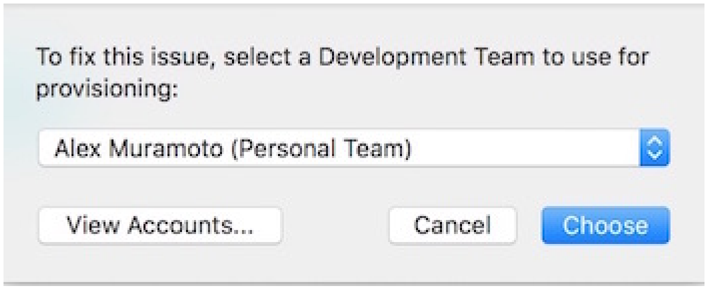


Failed to code sign "testApp".

No provisioning profiles with a valid signing identity (i.e. certificate and private key pair) matching the bundle identifier "com.ionicframework.testapp594195" were found.

Xcode can attempt to fix this issue. This will reset your code signing and provisioning settings to recommended values and resolve issues with signing identities and provisioning profiles.

Click the 'Fix Issue' button, then select your 'Personal Team' profile.

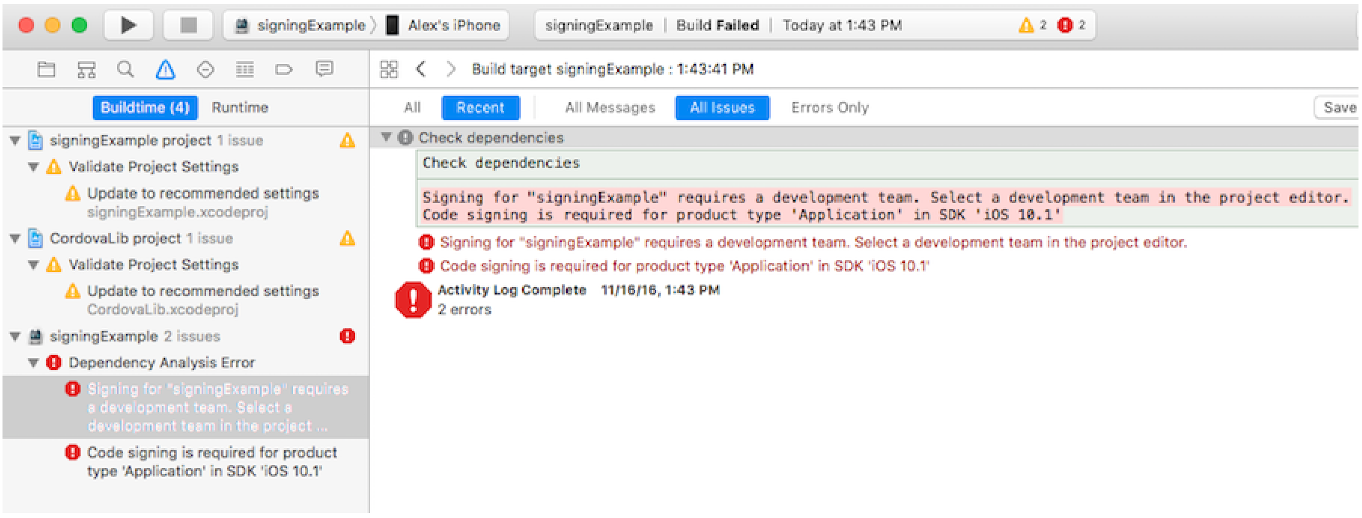


To fix this issue, select a Development Team to use for provisioning:

Alex Muramoto (Personal Team)

Xcode 8

If you are running Xcode 8, the code signing error will appear as a buildtime error, rather than as a pop-up:



signingExample project 1 issue
Validate Project Settings
Update to recommended settings
signingExample.xcodeproj
CordovaLib project 1 issue
Validate Project Settings
Update to recommended settings
CordovaLib.xcodeproj
signingExample 2 issues
Dependency Analysis Error
Signing for "signingExample" requires a development team. Select a development team in the project ...
Code signing is required for product type 'Application' in SDK 'iOS 10.1'

Build target signingExample : 1:43:41 PM

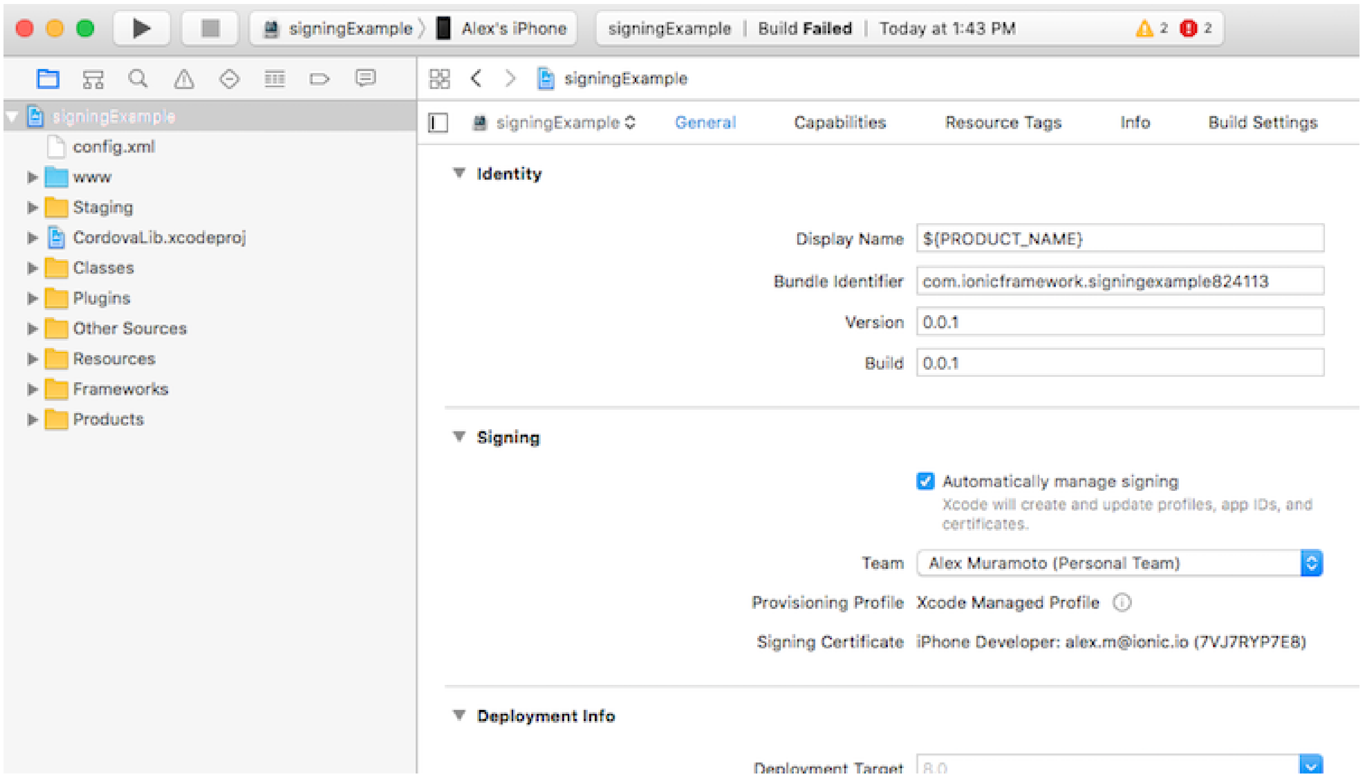
```
Check dependencies

Signing for "signingExample" requires a development team. Select a development team in the project editor.
Code signing is required for product type 'Application' in SDK 'iOS 10.1'
```

Signing for "signingExample" requires a development team. Select a development team in the project editor.
Code signing is required for product type 'Application' in SDK 'iOS 10.1'
Activity Log Complete 11/16/16, 1:43 PM
2 errors

To select the certificate to sign your app with, do the following:

1. Go to the 'Project Editor' by clicking the name or your project in the 'Project Navigator'
2. Select the 'General' section
3. Select the team associate with your signing certificate from the 'Team' dropdown in the 'Signing' section



Identity

Display Name	${PRODUCT_NAME}
Bundle Identifier	com.ionicframework.signingexample824113
Version	0.0.1
Build	0.0.1

Signing

Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

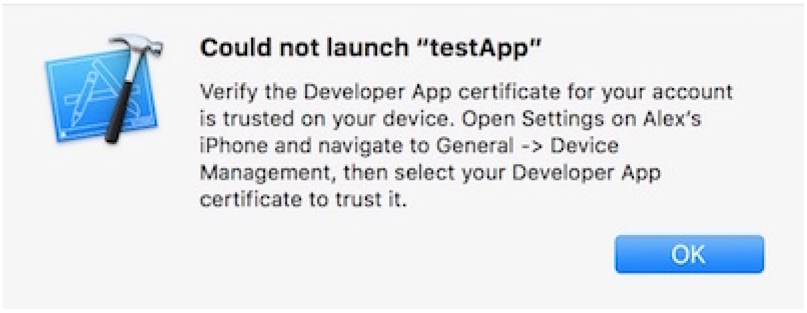
Team	Alex Muramoto (Personal Team)
Provisioning Profile	Xcode Managed Profile
Signing Certificate	iPhone Developer: alex.m@ionic.io (7VJ7RYP7E8)

Deployment Info

Deployment Target 8.0

Trusting the Certificate

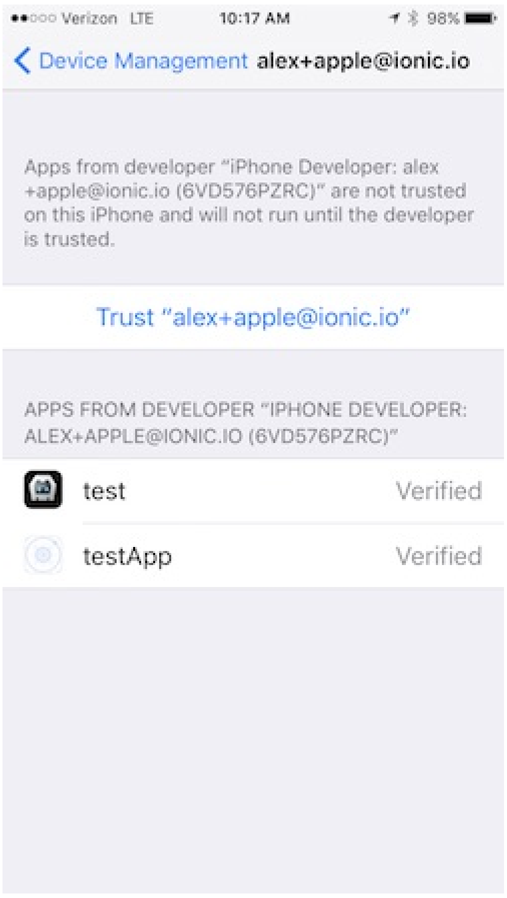
Once you've code signed your app, you should get a launch error that looks like this. On Xcode 7 and below you'll see this automatically. On Xcode 8 it will appear the next time you try to run the app:


Could not launch "testApp"

Verify the Developer App certificate for your account is trusted on your device. Open Settings on Alex's iPhone and navigate to General -> Device Management, then select your Developer App certificate to trust it.

To get past this, we have to tell our iOS device to trust the certificate we code signed our app with:

1. Open the 'Settings' app on your iOS device
2. Go to 'General > Device Management'. You'll see the email address associated with the Apple ID or Apple Developer account you used to code sign your app.
3. Tap the email address
4. Tap 'Trust <your_email>':


●●○○○ Verizon LTE 10:17 AM 98%
Device Management alex+apple@ionic.io

Apps from developer "iPhone Developer: alex +apple@ionic.io (6VD576PZRC)" are not trusted on this iPhone and will not run until the developer is trusted.

Trust "alex+apple@ionic.io"

APPS FROM DEVELOPER "IPHONE DEVELOPER: ALEX+APPLE@IONIC.IO (6VD576PZRC)"

test Verified

testApp Verified

Now, go back to Xcode and hit that play button or run `ionic cordova run ios --device` from the command line to install and launch your app on your iOS device.