

Setting up the development environment

Development OS: MacOS

Target OS: iOS

Installing dependencies

You will need Node, Watchman, the React Native command line interface, and Xcode.

While you can use any editor of your choice to develop your app, you will need to install Xcode in order to set up the necessary tooling to build your React Native app for iOS.

Node & Watchman

We recommend installing Node and Watchman using [Homebrew](#). Run the following commands in a Terminal after installing Homebrew:

```
brew install node  
brew install watchman
```

If you have already installed Node on your system, make sure it is Node 10 or newer.

[Watchman](#) is a tool by Facebook for watching changes in the filesystem. It is highly recommended you install it for better performance.

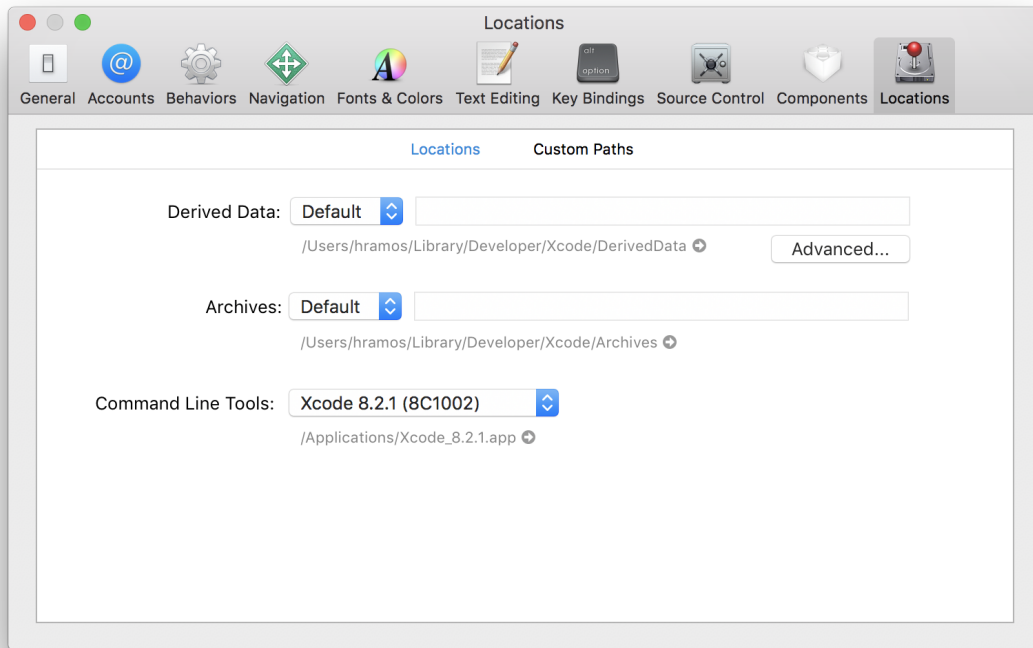
Xcode & CocoaPods

The easiest way to install Xcode is via the [Mac App Store](#). Installing Xcode will also install the iOS Simulator and all the necessary tools to build your iOS app.

If you have already installed Xcode on your system, make sure it is version 9.4 or newer.

Command Line Tools

You will also need to install the Xcode Command Line Tools. Open Xcode, then choose "Preferences..." from the Xcode menu. Go to the Locations panel and install the tools by selecting the most recent version in the Command Line Tools dropdown.



Installing an iOS Simulator in Xcode

To install a simulator, open **Xcode > Preferences...** and select the **Components** tab. Select a simulator with the corresponding version of iOS you wish to use.

CocoaPods

[CocoaPods](#) is built with Ruby and it will be installable with the default Ruby available on macOS. You can use a Ruby Version manager, however we recommend that you use the standard Ruby available on macOS unless you know what you're doing.

Using the default Ruby install will require you to use `sudo` when installing gems. (This is only an issue for the duration of the gem installation, though.)

```
sudo gem install cocoapods
```

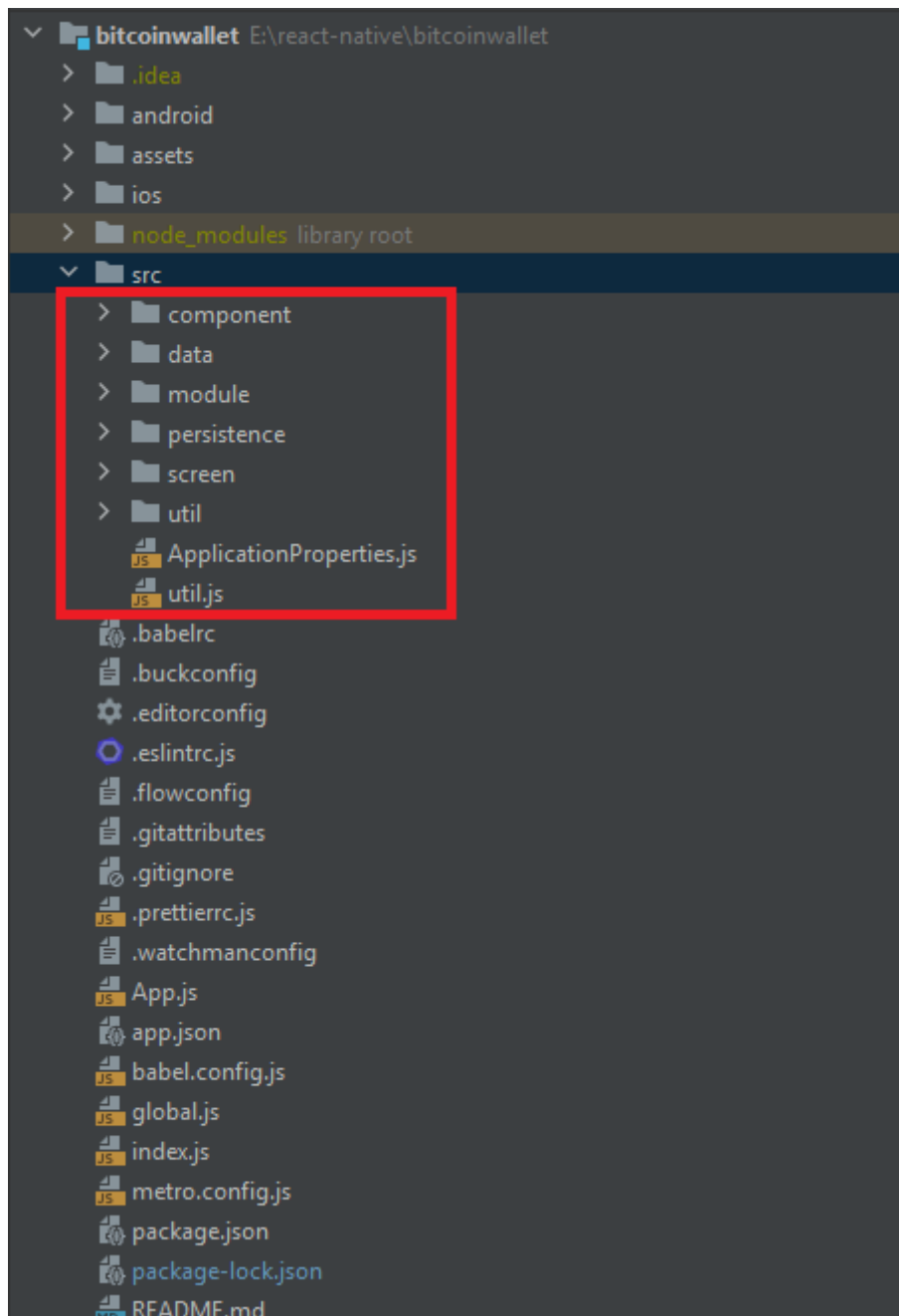
For more information, please visit [CocoaPods Getting Started guide](#).

React Native Command Line Interface

React Native has a built-in command line interface. Rather than install and manage a specific version of the CLI globally, we recommend you access the current version at runtime using `npx`, which ships with Node.js. With `npx react-native <command>`, the current stable version of the CLI will be downloaded and executed at the time the command is run.

Import BitcoinWalletproject

I'm going to use IntelliJ IDEA as the IDE. You can choose which IDE as your preferable IDE as well



Run BitcoinWalletProject

Then run the following command to start our application

```
$ cd ios
$ pod install --repo-update
$ cd ..
npx react-native run-ios
```

After a while, Our application will run on your connected iOS device.

11:45



Home



Contact



Marketplace



About Us



Language



Powered by @LM

testnet

Wallets

Wallet 2

0.00119

Latest Transaction

Never

Marketplace



Bitrefill
Buy vouchers



silentburner
Get global burner numbers



Stekki
Earn free Bitcoin



Insms
Send text messages



Zigzag
Exchange

Publishing to App Store

You have built a great app using React Native, and you are now itching to release it in the App Store. The process is the same as any other native iOS app, with some additional considerations to take into account.

1. Enable App Transport Security

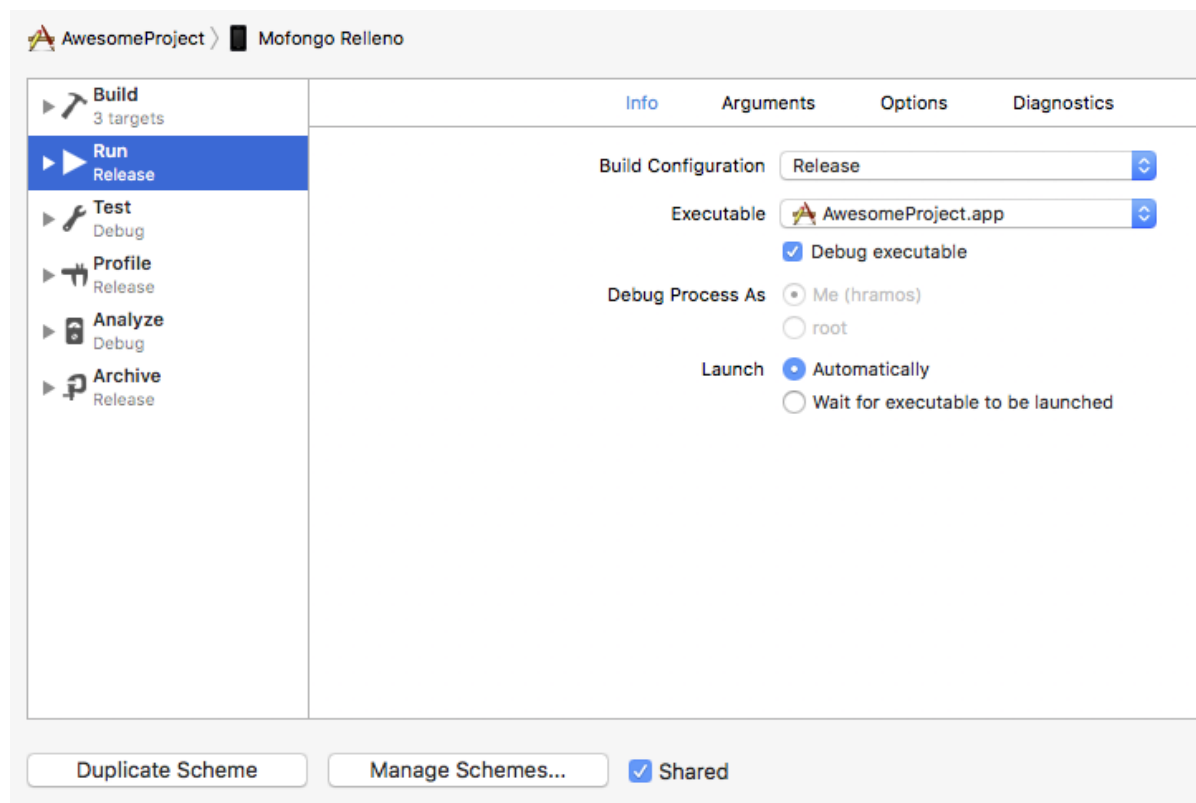
App Transport Security is a security feature introduced in iOS 9 that rejects all HTTP requests that are not sent over HTTPS. This can result in HTTP traffic being blocked, including the developer React Native server. ATS is disabled for `localhost` by default in React Native projects in order to make development easier.

You should re-enable ATS prior to building your app for production by removing the `localhost` entry from the `NSExceptionDomains` dictionary and setting `NSAllowsArbitraryLoads` to `false` in your `Info.plist` file in the `ios/` folder. You can also re-enable ATS from within Xcode by opening your target properties under the Info pane and editing the App Transport Security Settings entry.

2. Configure release scheme

Building an app for distribution in the App Store requires using the `Release` scheme in Xcode. Apps built for `Release` will automatically disable the in-app Developer menu, which will prevent your users from inadvertently accessing the menu in production. It will also bundle the JavaScript locally, so you can put the app on a device and test whilst not connected to the computer.


To configure your app to be built using the `Release` scheme, go to **Product** → **Scheme** → **Edit Scheme**. Select the **Run** tab in the sidebar, then set the Build Configuration dropdown to `Release`.



As your App Bundle grows in size, you may start to see a blank screen flash between your splash screen and the display of your root application view. If this is the case, you can add the following code to `AppDelegate.m` in order to keep your splash screen displayed during the transition.

```
// Place this code after "[self.window makeKeyAndVisible]" and before "return YES;"
    UIStoryboard *sb = [UINavigationController storyboardWithName:@"LaunchScreen"
bundle:nil];
    UINavigationController *vc = [sb instantiateInitialViewController];
    rootView.loadingView = vc.view;
```

3. Build app for release

You can now build your app for release by tapping  or selecting **Product** → **Build** from the menu bar. Once built for release, you'll be able to distribute the app to beta testers and submit the app to the App Store.

That's it!

Congratulations! You've successfully run our BitcoinWalletApp.