

Build

Build Environment

Cordova Version	Android	iOS
Cordova 10.0	AndroidSDK version : 29	Xcode version : 11.3 / 12.2
Cordova 9.0	AndroidSDK version : 28	Xcode version : 10.1 / 10.2 / 10.2.1 / 11.3
Cordova 7.1	AndroidSDK version : 26	Xcode version : 9 / 10.1
<ul style="list-style-type: none">• Building for iOS• Building for Android• Building for Windows• Building for Electron• Building for PWA• Common Build and Application Uploader Errors• Build Environment Settings• Build History		

Building for Android

Types of Build

In Monaca, Android app has two types of build: debug version and release version. The differences between these types of build are as follows:

Types of Build	Description	Installation
Debug Build	An unsigned package which cannot be distributed in the market	<ul style="list-style-type: none">• QR Code• Network Install• Sideload
Release Build	A signed package with the developer's code sign which can be distributed in the market	<ul style="list-style-type: none">• Sideload• Google Play Store and other eligible markets

Sideload typically refers to media file transfer to a mobile device via USB, Bluetooth, WiFi or by writing to a memory card for insertion into the mobile device. When referring to Android apps, "sideloading" typically means installing an application package in APK format onto an Android device without going through the market.

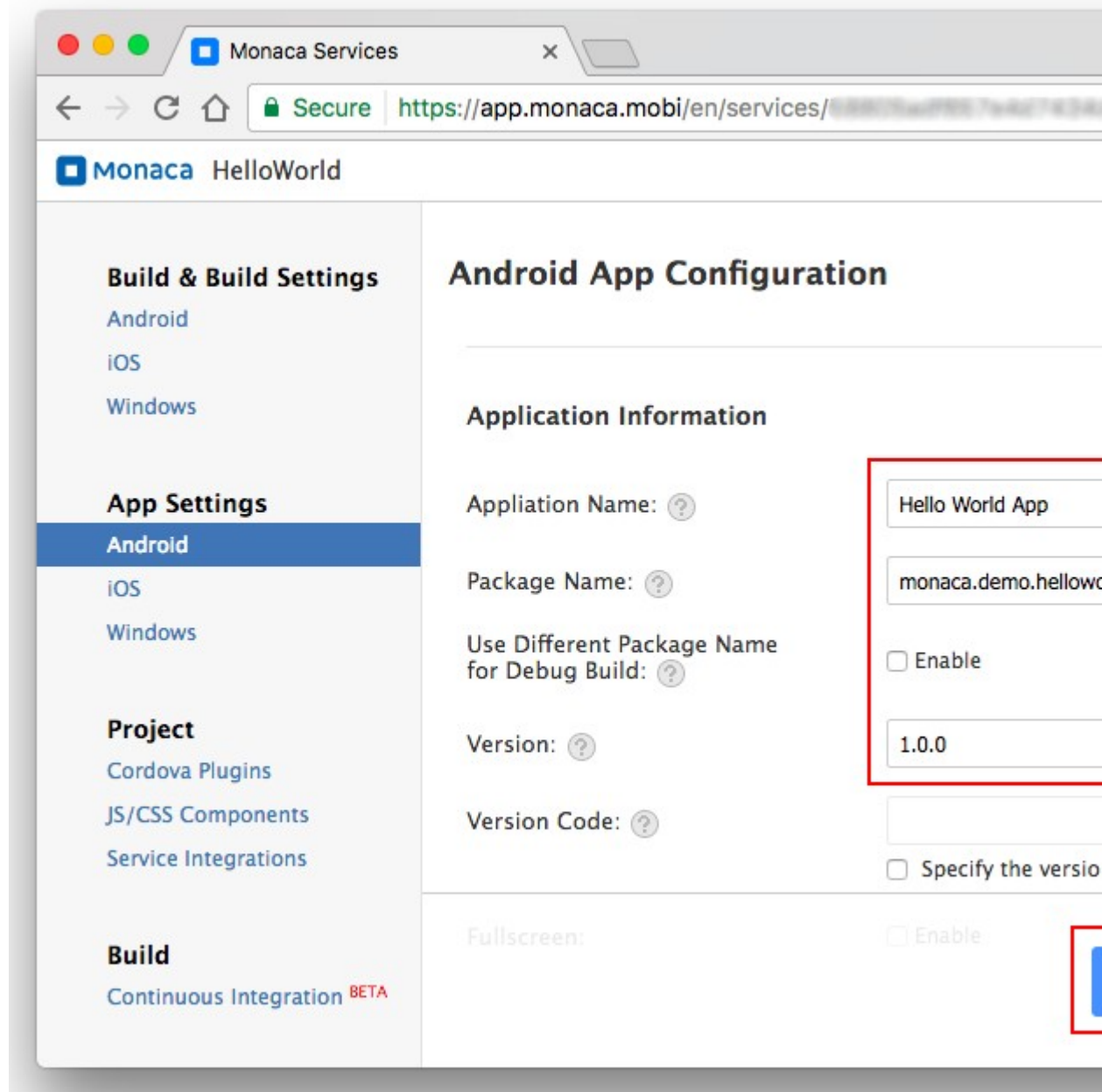
Step 1: Configure Android App

1. From the Monaca Cloud IDE menu, go to **Configure** → **App Settings for Android**.
2. Fill in the necessary information of your app:

- General settings:

Application Name	A name representing your app publicly such as in the Market
Package Name	A unique name which will be used when uploading to the Android Market. It is recommended to use reverse-domain style (for example, io.monaca.app_name) for App ID. Only alphanumeric characters, periods (at least one period must be used) and underscore are allowed. Each segment should be separated by a period and started with an alphabetic character.
Use Different Package Name for Debug Build	If enable, the package name of the release-built and debug-built apps are different. In other words, the package name of debug-built app will have <code>.debug</code> extension, and the one for project debugger will have <code>.debugger</code> extension. However, this option is disabled by default because it made some plugins impossible to be debugged due to the fact that they are tied to exact package names (eg. in-app purchase).
Version	The version number of your app. A version number consists of only numbers separated by dots (for example, 1.0.0).
Version Code	An internal version number of your app, relative to other versions. The value must be integer, so that the applications can programmatically evaluate it for an upgrade.
Fullscreen	This option is only available with the Cordova 3.5 and later.

If enable, your app will be run in a fullscreen mode which hide the status bar.



- Misc: various settings regarding your Android app such as:

Allowed URL	*	Specify URL(s) which can be accessed from your app. If set to *, you can access all domains from your app.
Keep Running	Enable	Enable this if you want Cordova to keep running in the background.
Disallow Overscroll	Enable	Enable this if you want to disable the glow when a user scrolls beyond the edge of the webview.
Screen Orientation	Default	You can also set the device's screen orientation when running your app as Landscape or Portrait.

After finishing the configurations, click Save.

Currently, when you update either iOS's App ID or Android's Package Name, both of them will change. In other words, they are configured to be the same. However, it is possible to

make them different. Please refer to [How to make iOS's App ID and Android's Package Name differently](#).

3. Set of adaptive icons

If you are using a monaca project with cordova 9.0 or later, you can set an adaptive icon for android 8.0 or later.

To set an adaptive icon, upload an image file under the `res` folder and add a `background` attribute and a `foreground` attribute to the `icon` tag of `config.xml`. If the previous `src` attribute is set, the value of the `src` attribute is not used.

If the os version does not support adaptive icons, the image set in the `foreground` attribute will be displayed. If the previous `src` attribute is set, the value of the `src` attribute takes precedence.

The following is an example of placing an image under `/res/android/icon/`.

```
<platform name="android">
  <icon density="ldpi" background="/res/android/icon/ldpi-
background.png" foreground="/res/android/icon/ldpi-foreground.png"/>
  <icon density="mdpi" background="/res/android/icon/mdpi-
background.png" foreground="/res/android/icon/mdpi-foreground.png"/>
  <icon density="hdpi" background="/res/android/icon/hdpi-
background.png" foreground="/res/android/icon/hdpi-foreground.png"/>
  <icon density="xhdpi" background="/res/android/icon/xhdpi-
background.png" foreground="/res/android/icon/xhdpi-foreground.png"/>
  <icon density="xxhdpi" background="/res/android/icon/xxhdpi-
background.png" foreground="/res/android/icon/xxhdpi-foreground.png"/>
  <icon density="xxxhdpi" background="/res/android/icon/xxxhdpi-
background.png" foreground="/res/android/icon/xxxhdpi-foreground.png"/>
</platform>
```

The following is an example of setting the `src` attribute.

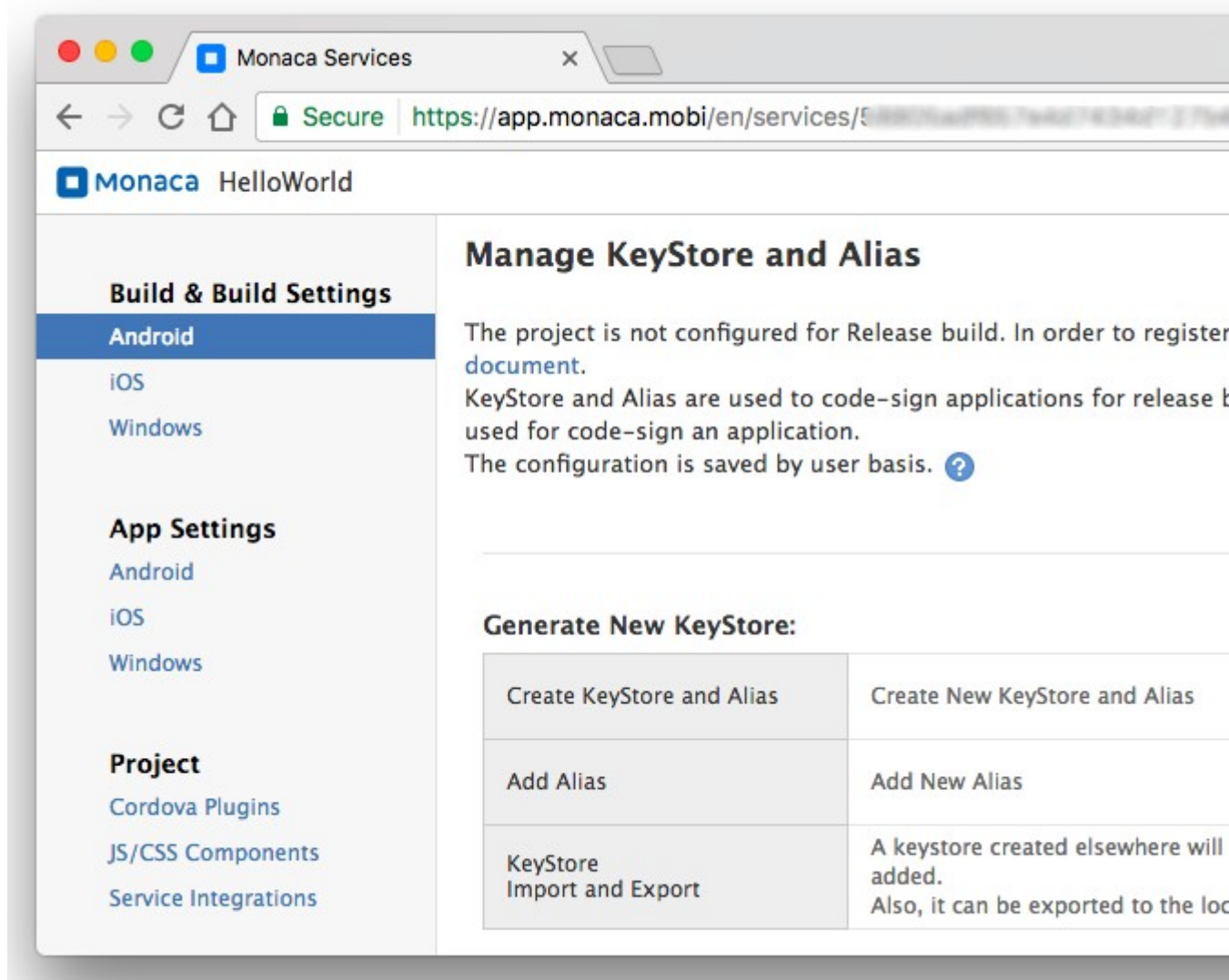
```
<platform name="android">
  <icon src="/res/android/icon/ldpi.png" density="ldpi"
background="/res/android/icon/ldpi-background.png"
foreground="/res/android/icon/ldpi-foreground.png"/>
  <icon src="/res/android/icon/mdpi.png" density="mdpi"
background="/res/android/icon/mdpi-background.png"
foreground="/res/android/icon/mdpi-foreground.png"/>
  <icon src="/res/android/icon/hdpi.png" density="hdpi"
background="/res/android/icon/hdpi-background.png"
foreground="/res/android/icon/hdpi-foreground.png"/>
  <icon src="/res/android/icon/xhdpi.png" density="xhdpi"
background="/res/android/icon/xhdpi-background.png"
foreground="/res/android/icon/xhdpi-foreground.png"/>
  <icon src="/res/android/icon/xxhdpi.png" density="xxhdpi"
background="/res/android/icon/xxhdpi-background.png"
foreground="/res/android/icon/xxhdpi-foreground.png"/>
  <icon src="/res/android/icon/xxxhdpi.png" density="xxxhdpi"
background="/res/android/icon/xxxhdpi-background.png"
foreground="/res/android/icon/xxxhdpi-foreground.png"/>
</platform>
```

Step 2: Configure Android Keystore

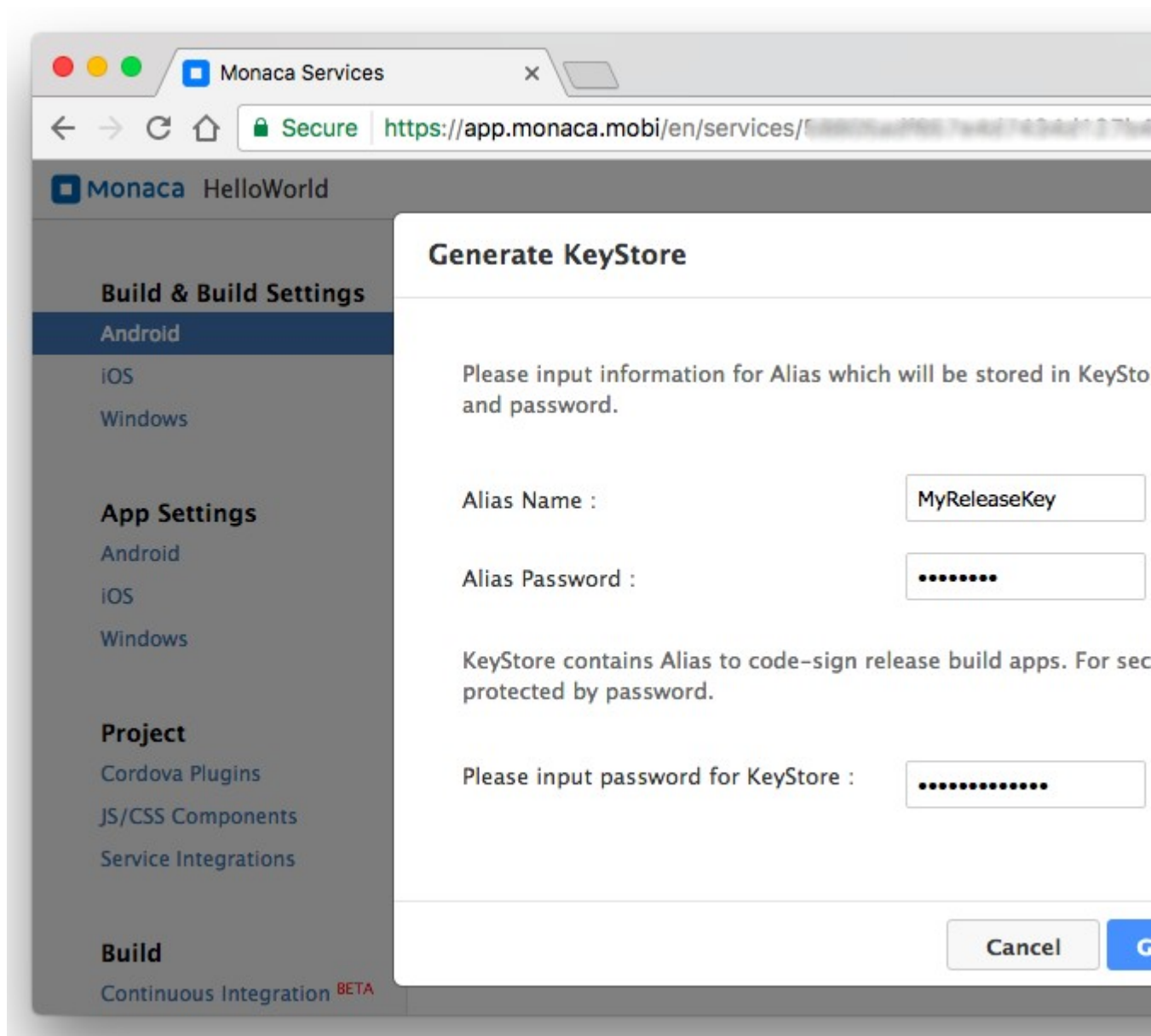
A keystore is a binary file that contains a set of private keys. A private key represents the entity to be identified with the app, such as a person or a company. A keystore is encrypted with a password and it cannot be restored if the password is lost. When a keystore is lost or it overwrites another keystore, it is impossible to use the same key to re-sign the signed package.

A keystore is required for the building of a release version for your Android app. In Monaca, you can either create a new keystore or import an existing one. In order to create a new keystore, please do as follows:

1. From the Monaca Cloud IDE menu, go to **Configure** → **Android KeyStore Settings**.
2. Then, Manage KeyStore and Alias page will appear.



3. Click on Clear and Generate New button. Then, the following screen will appear:



4. Fill in the necessary information as shown in the above screen such as:

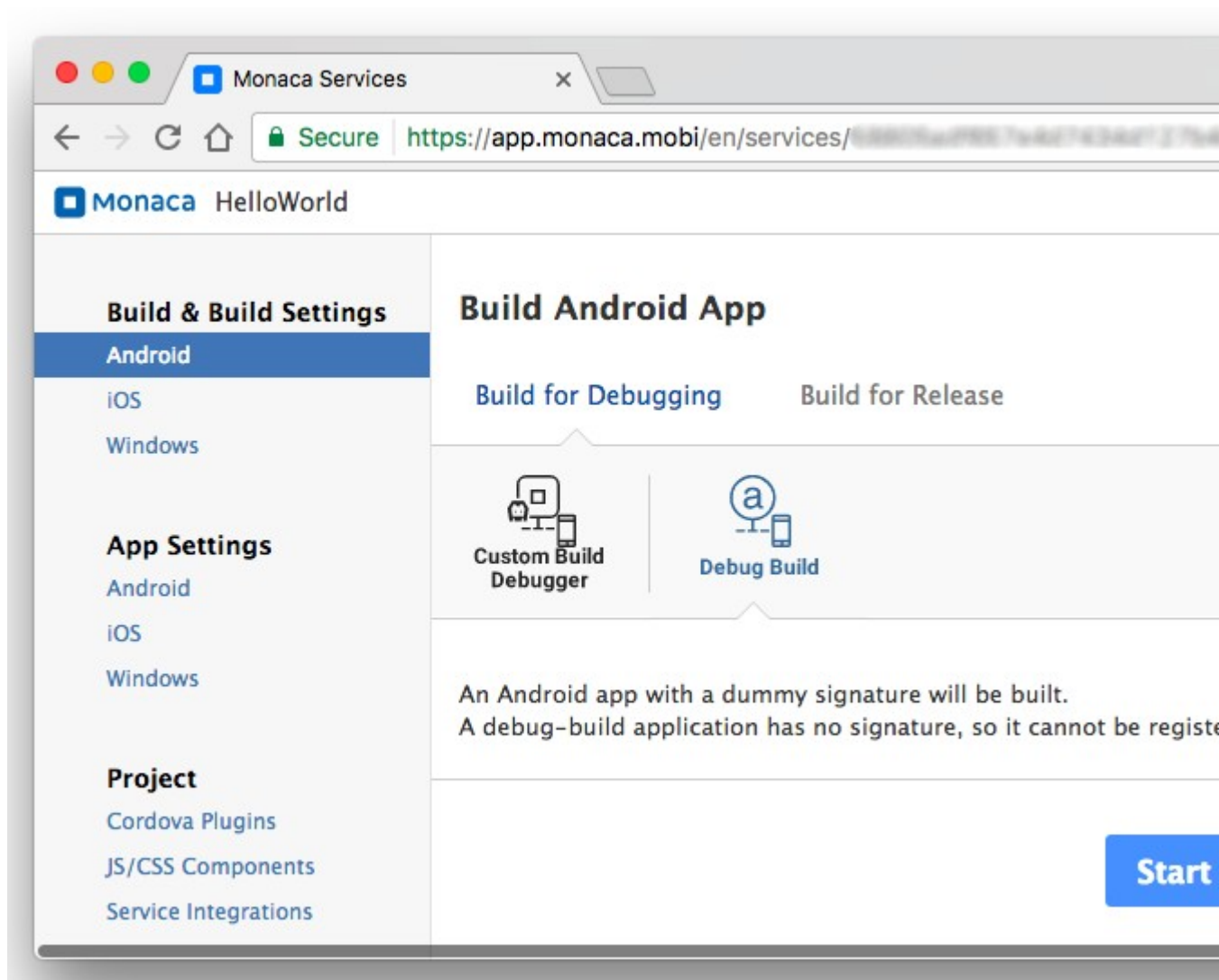
- **Alias:** a name representing a private key that you will use later when signing your app. Multiple aliases can be stored within one keystore.
- **Password:** a password for the private key (alias).
- **Password of the keystore:** a password for the keystore. You will need this password when importing this keystore.

5. Then, click on Generate Keystore and Alias button to Generate the keystore.

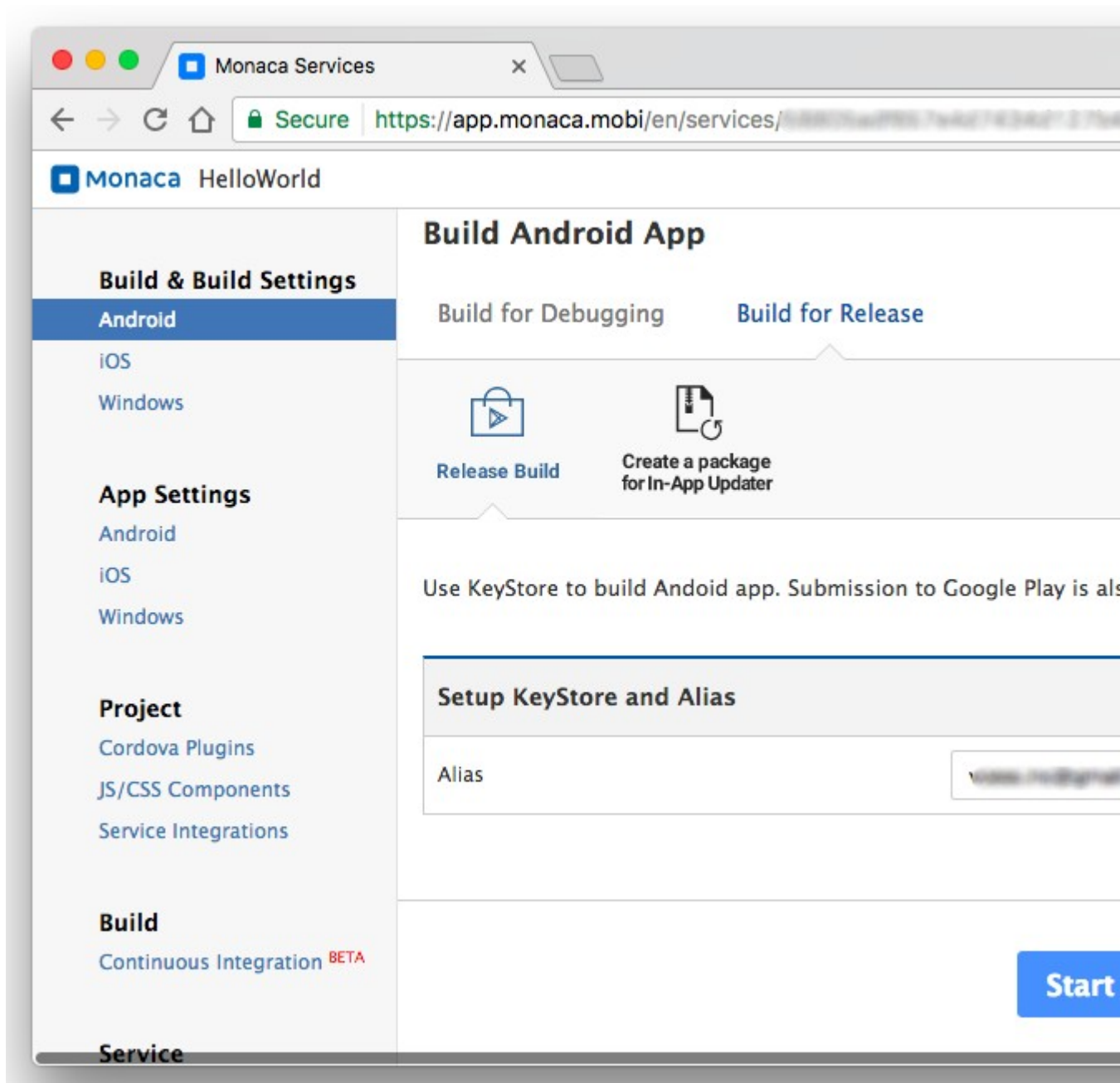
When a keystore is lost, it is impossible to use the same key to re-sign the signed package. Therefore, always back up and keep the keystore which is used to sign application(s). Use the Export button to download your keystore.

Step 3: Start Building

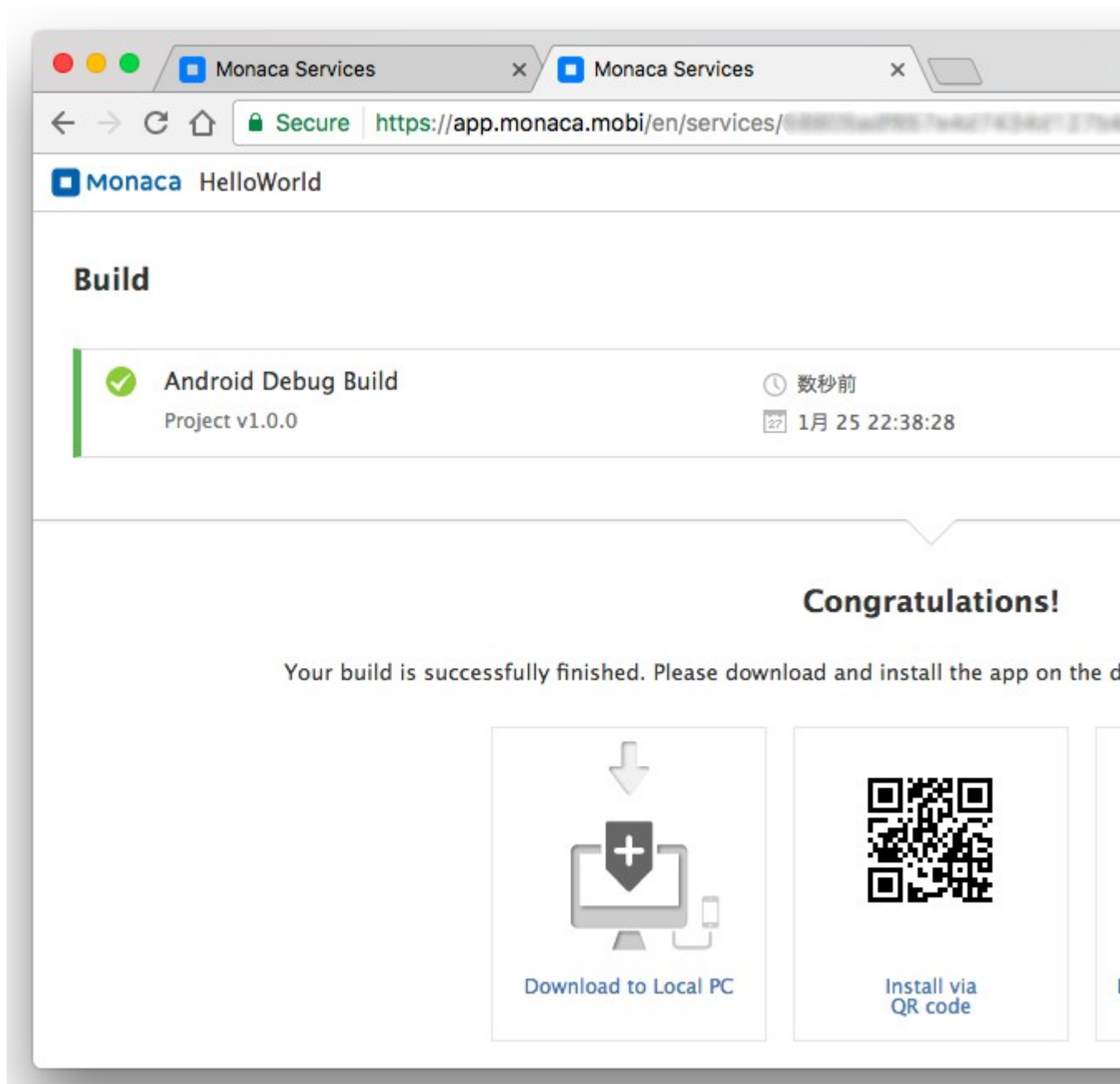
1. From the Monaca Cloud IDE menu, go to **Build** → **Build App for Android**.
2. Select appropriate type of build you want and click **Start Build**.



3. If you choose **Release Build**, you will also need to select an alias to sign your package before start building.



4. It may take several minutes for the build to complete. Please wait. Once the build is completed, your built app is ready to be installed/downloaded. See below screenshot as an example:



See Also:

- [Building for iOS](#)
- [Building for Windows](#)
- [Google Play Distribution](#)
- [Build History](#)

Building for Electron

- [Building an Electron Windows Application](#)
- [Building an Electron macOS Application](#)
- [Building an Electron Linux Application](#)

Building for Linux Application

Supported environment: Cordova 9.0

Types of Build

In Monaca, Electron application has two types of build: debug version and release version. The differences between these types of build are as follows:

Types of Build	Description
Debug Build	DevTools are shown by default when the application is launched. It can be closed or opened manually.
Release Build	There are no DevTools. The built package is unsigned.

Step 1: Configure App

1. From the Monaca Cloud IDE menu, go to **Configure → App Settings for Linux**.
2. Fill in the necessary information of your app:

- Application information:

Application Name	The application name.
App ID	A unique name which will be used when uploading to the Store. It is recommended to use reverse-domain style (for example, io.monaca.app_name) for App ID. Only alphanumeric characters, periods (at least one period must be used) and underscore are allowed. Each segment should be separated by a period and started with an alphabetic character.
Version Number	The version number of your app. A version number consist of only number seperated by dots (for example, 1.0.0)
Application Description	The description of your application.

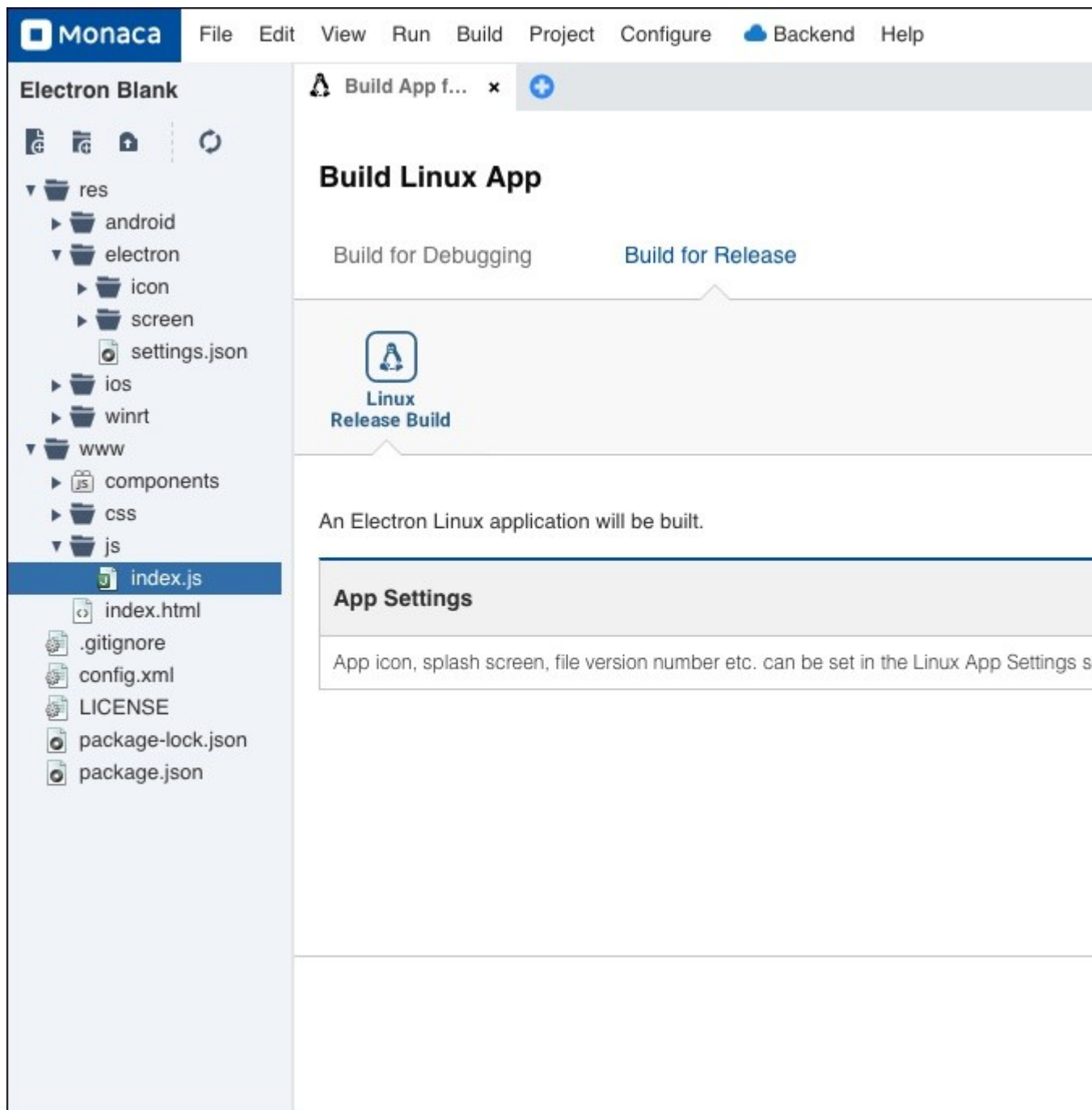
- Application Icon

You can set the application icon. PNG format is supported. Icon should be at least 512x512 pixels to work across all operating systems.

3. After finishing the configurations, click Save.

Step 2: Start Building

1. From the Monaca Cloud IDE menu, go to Build → Build App for Linux.
2. Select appropriate type of build you want and click Start Build button.




3. It may take several minutes for the build to complete. Please wait. Once the build is completed, your built application is ready to be downloaded.

Electron Blank

▶  node_modules

▼  res

▶  android

▼  electron

▶  icon


▶  screen

 settings.json

▶  ios

▶  winrt

▼  www

▶  components

▶  css


▼  js

 index.js

 index.html

 manifest.json

 .gitignore

 config.xml

 LICENSE

**Build Results** x**Build**

Step 3: Install The App

1. Extract the downloaded zip file.
2. The name of executable file is the App ID filename specified in the App Configuration page. Open terminal and execute the file.

Monaca

FileEditViewRunBuildProjectConfigureBackendHelp

Electron Blank

node_modules

res

android

electron

icon

screen

settings.json

ios

winrt

www

components

css

js

index.js

index.html

manifest.json

.gitignore

config.xml

LICENSE

package-lock.json

package.json

App Setting... x

Linux App Configuration

Please input necessary configurations for building Linux application. This co

Application Information

Application Name: ?

App ID: ?

Version Number: ?

Application Description: ?

Electron Minimum

com.example.electronminimum

1.0.0

Made with Monaca (<http://monaca>)



Tested Environment

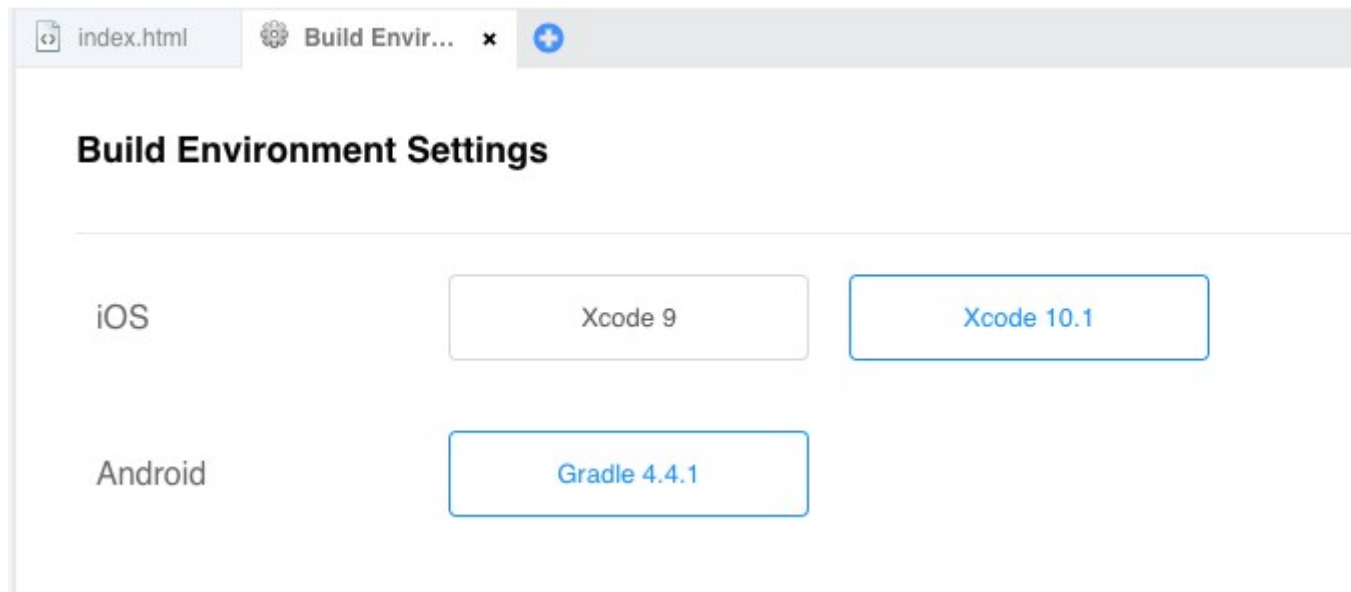
- Ubuntu 16.04.1

Build Environment Settings

You can set environment settings of building the application. The settings are registered per project. The build environment settings can be used from Cordova 7.1 or later.

Configure Build Environment

1. From the Monaca Cloud IDE menu, go to **Build** → **Build Environment Settings** .
2. Set environment information to be used for build app.



3. After finishing the configurations, click Save.