

# Tutorial

Monaca Cloud IDE provides a browser-based development environment as a service. Right from your Web browser, all of your Cordova development is done without any setup. Along with the [Monaca Debugger](#), Live Preview (built-in function in Monaca Cloud IDE) allows you to easily check the progress of your apps during development.

Before getting started with this tutorial, you will need the following:

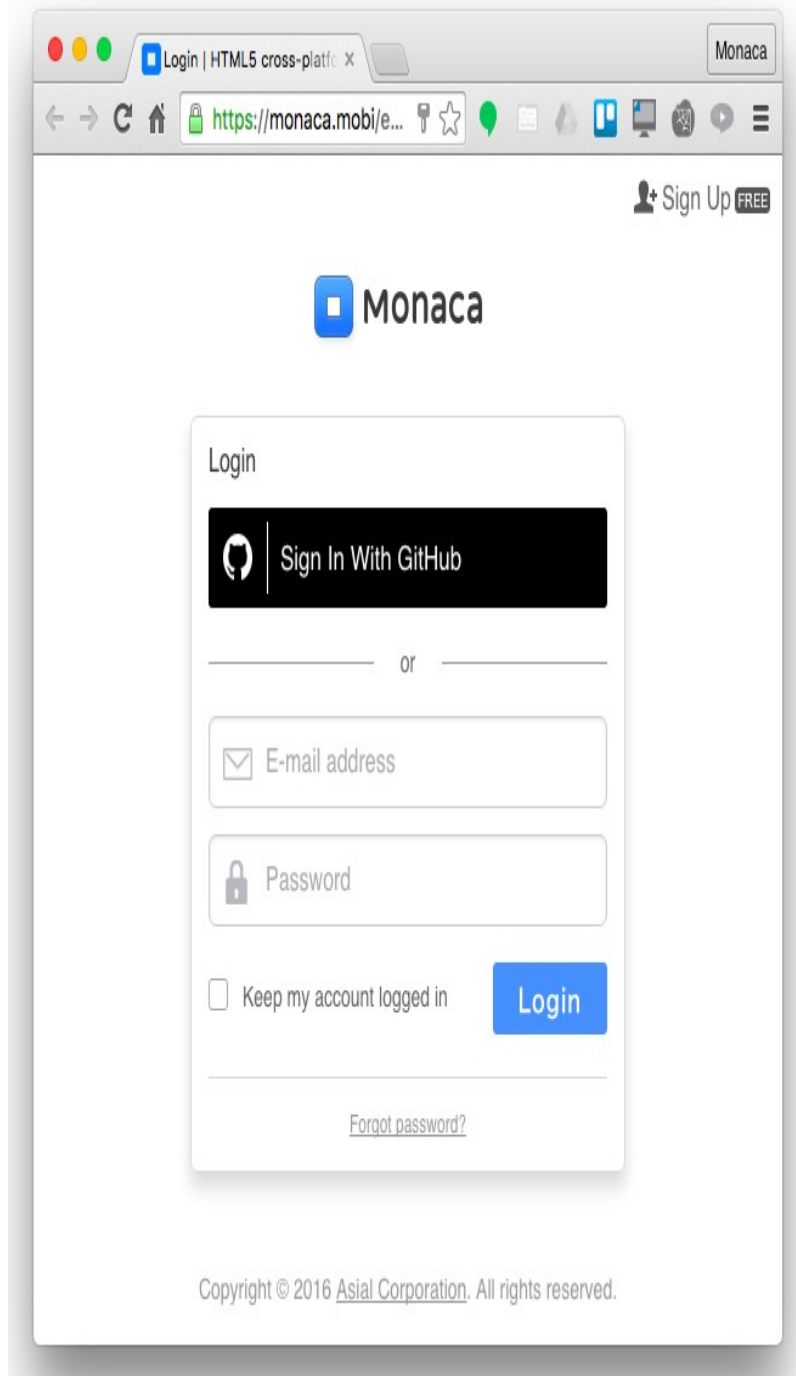
- a computer with the Internet connection
- [optional] a smart mobile device (either iOS or Android) if you want to test on a real device

- 
- [Part 1: Starting a Project](#)
  - [Part 2: Running Monaca Debugger with Monaca Cloud IDE](#)
  - [Part 3: Building Monaca App](#)
  - [Part 4: Publishing Monaca App](#)

# Part 1: Starting a Project

## Step 1: Logging into Monaca

1. Login to [Monaca](#).
2. Enter your Monaca account information.



## Step 2: Creating a New Project

After successfully signing in, you will be redirected to the Monaca Dashboard. You are now ready to create your first Monaca project! Let's do the following:

1. From the Monaca Dashboard, you can choose Create Project or Import to create a new Monaca project. If you select **Create Project**, you can create a new project based on various templates or from scratch. On the other hand, if you choose **Import**, you can create a new project from an existing Cordova/Monaca project in a number of different ways such as:

- import from URL (zip file)
- upload a project (zip file)
- import from Git repository (see [Git SSH Integration](#))
- or import from your existing GitHub repository (see [GitHub Integration](#)).

## Import Project

**Project Name \***

**Description**

**Import Method**

☐ **Import from URL**

☐ **Upload Project Package**

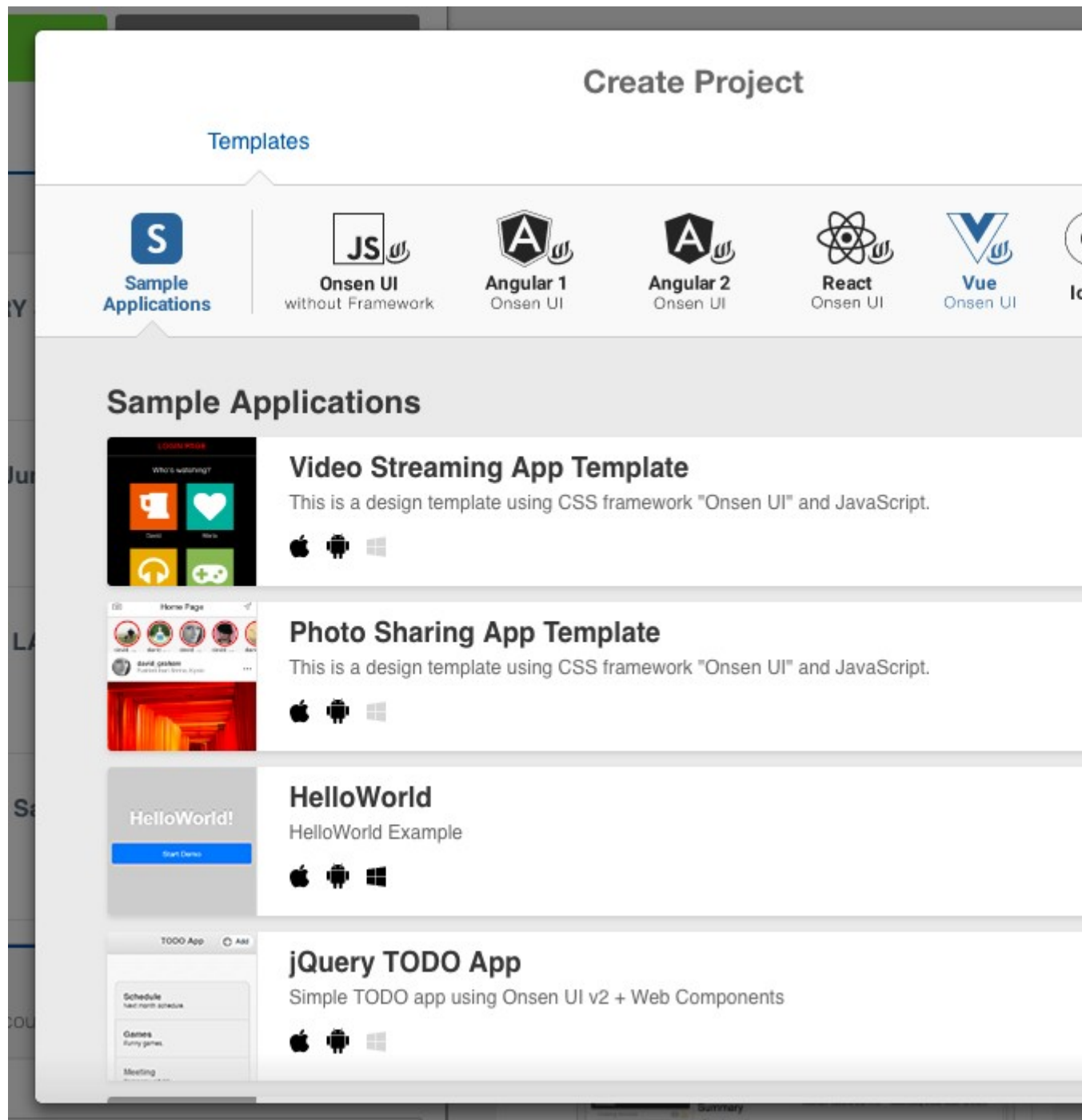
No file chosen

☐ **Import from Git Repository**

☐ **Import from GitHub Repository**

**Import**

**Cancel**



2. For this tutorial, we will choose the **Create Project** option. In the **Create Project** dialog, under **Sample Applications**, select the **jQuery TODO App** template by clicking on its **Create New** button.
3. Simply click on the **Make Project** button.



**New Project**

**Project Name \***

jQuery TODO App

**Description**

Make Project

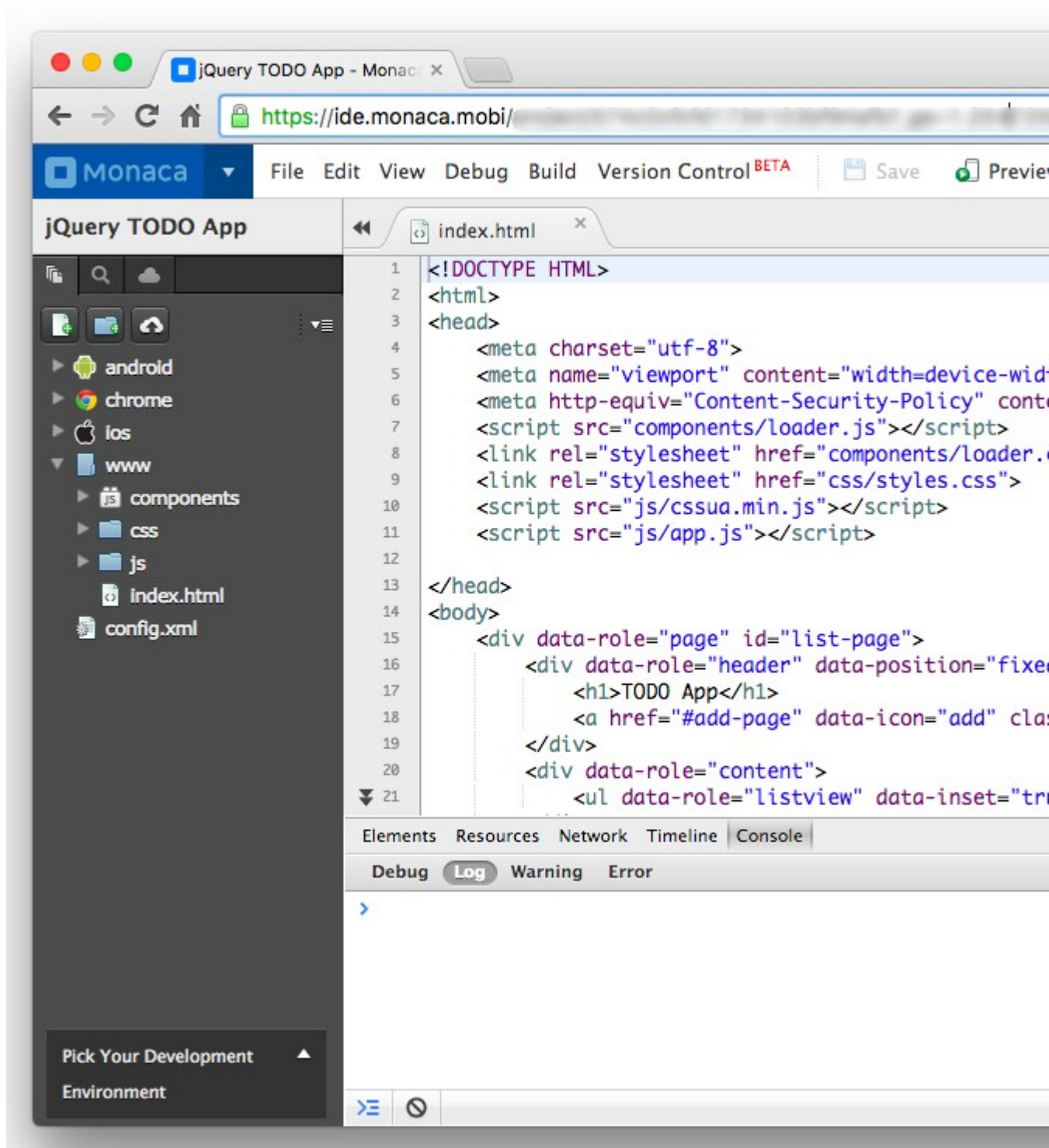
4. You will then see your newly created project listed under the **On Line** tab of the Dashboard.

### Step 3: Previewing a Project

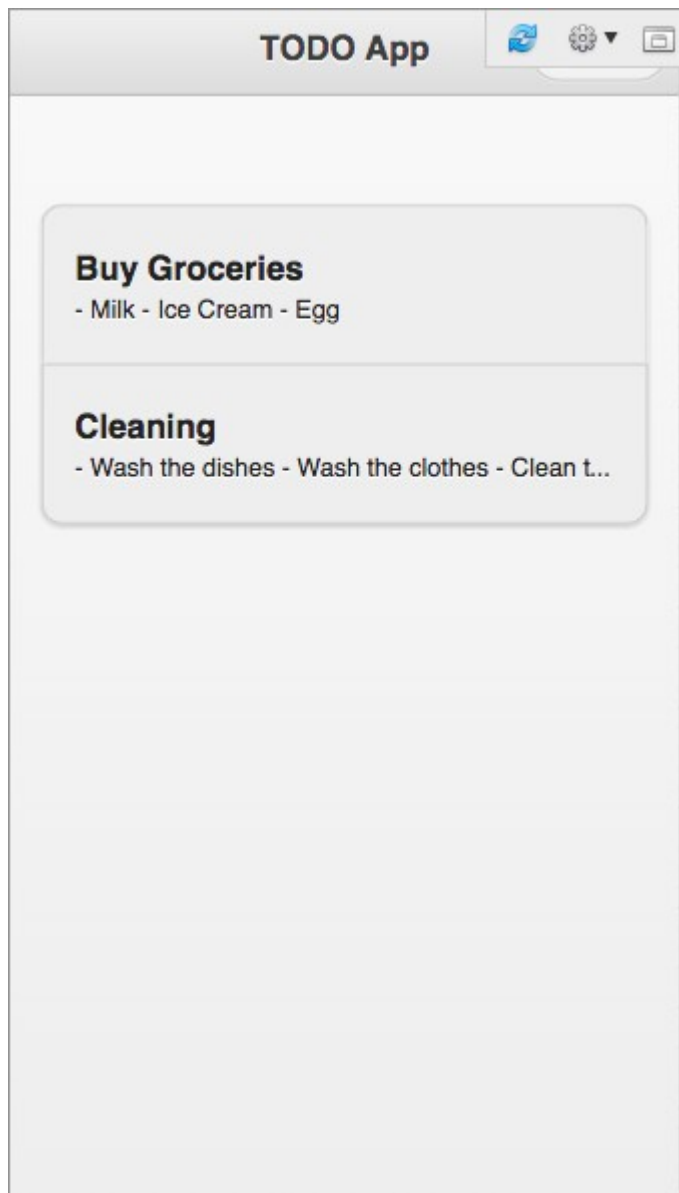
Monaca Cloud IDE allows you to preview your project through a Preview window without any real devices.

Some functionalities of applications might not be previewed properly because the Preview window has several limitations on Ajax requests, Cordova plugin APIs, etc. For the full list of limitations, please refer to [Usage and Limitation of Live Preview](#).

1. From the Monaca Dashboard, open the project we just created. Then, Monaca Cloud IDE will be opened. The Preview window is at the right side of the IDE by default.



2. Try adding some TODO items in the Preview window.



## Step 4: Editing a Project

All editable files are listed under `www/` folder.

1. From the File Tree panel, choose a file to edit. Let's make some changes in the `index.html` file using the code editor. For example, change the title of the app to something else.
2. Save the changes, then you will be able to see the updates instantly in the Preview window. Feel free to edit the project as you wish. For a more detailed explanation about this template, please refer to [jQuery TODO App](#).

For more information regarding the code editor in Monaca Cloud IDE, please refer to [Code Editor](#).

**Next:**

- [Part 2: Running Monaca Debugger with Monaca Cloud IDE](#)



## Part 2: Running Monaca Debugger with Monaca Cloud IDE

[Monaca Debugger](#) is a powerful application for testing and debugging your Monaca applications on real devices in real-time.

When developing Monaca apps in the Cloud IDE, all changes made to your project file(s) will be synced with your Monaca Debugger in real-time as soon as you save them.

### Before Getting Started

Please install Monaca Debugger on your device.



Please refer to [Monaca Debugger Installation](#) for other platforms.

### Step 1: Running a Project on Monaca Debugger

1. Launch Monaca Debugger app and sign in with your Monaca account. Make sure you are using the same account for Monaca Cloud IDE.



Debugger for iOS



demo@monaca.mobi



••••••••



Log In

2. The list of projects will appear. All Monaca Cloud IDE projects are listed under **Monaca.io Projects**. To run a project, simply tap on the project name in Monaca Debugger or select **Run** → **Run on Device** in Monaca Cloud IDE.



## Local Projects

No local project is found. This is because no local computer is connected and providing local projects.

▶ [See the instructions for local development.](#)

## Monaca.io Projects



Memo Application

 Not synced yet



3. Your project should now be running in the debugger.
4. Go ahead and try out the various features of the demo app!

## Monaca Memo

+ Add

2016/2/5

This is my second memo.



2016/2/5

This is my first memo.



To go back to the Project List screen, go to Debugger Menu and tap on the *Back* button.

Monaca Memo

+ Add

No memo found

Debugger Menu

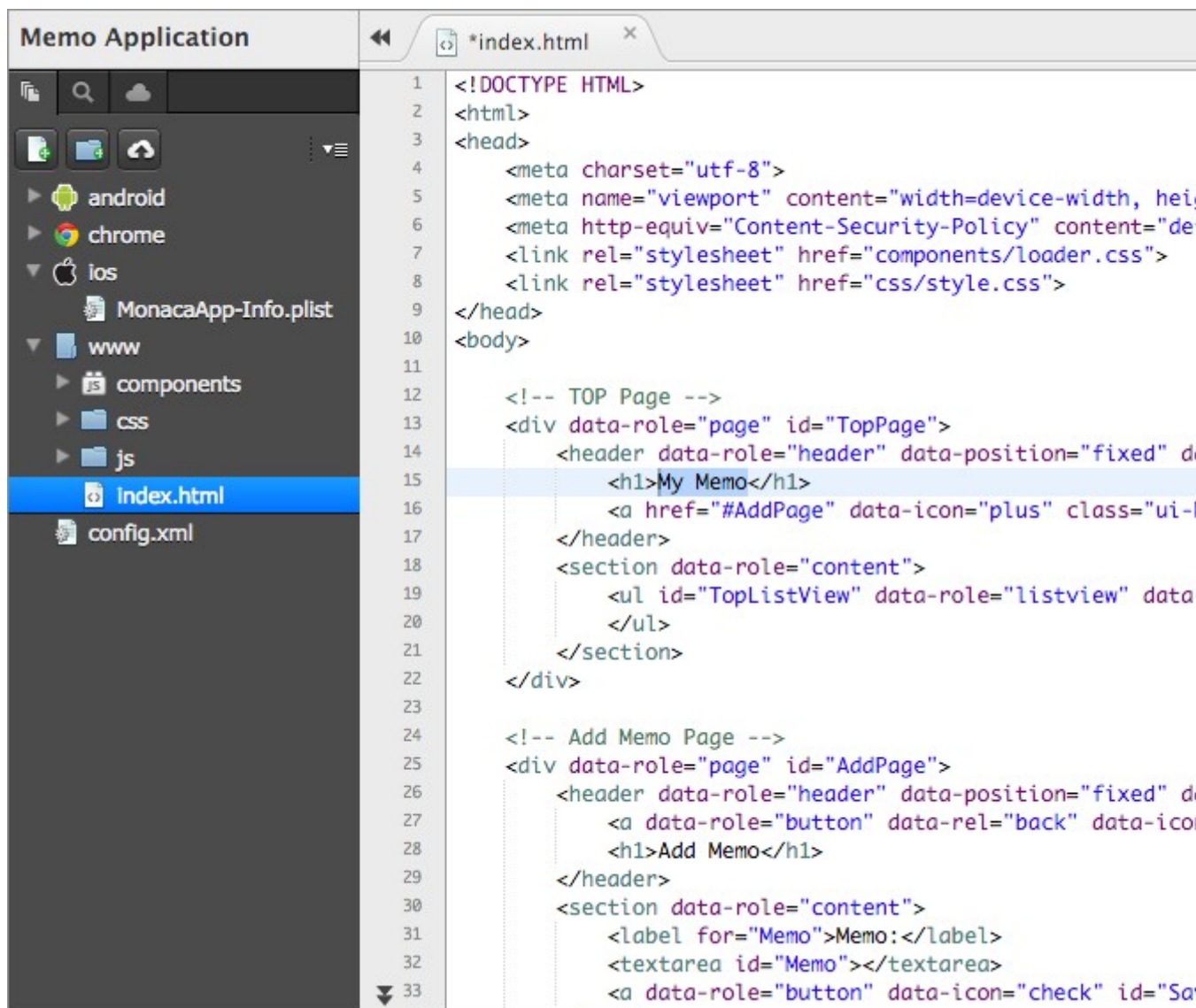




## Step 2: Real-time Updates between Monaca Cloud IDE and Debugger

By now, you should be able to get your Monaca project running on the debugger. Next, let's try to make changes to this project and see how it is reflected in the debugger.

1. Run the project on the debugger.
2. On Monaca Cloud IDE, let's open `index.html` and change the title of the front page from Monaca Memo to My Memo. Then, save the change.



3. As soon as you save your new changes, Monaca Debugger will automatically refresh itself. (You can also tap on *Reload* button to manually retrieve the latest updates of your app, in case the changes are not reflected.)

## My Memo

+ Add

2016/2/5

This is my second memo.



2016/2/5

This is my first memo.



Please refer to [Debugger Functionalities](#) to explore the other functions provided by Monaca Debugger.

You can also use USB debugging with Monaca CLI. Please refer to [USB Debugging with Monaca](#).

That's it! That's how easy it is to use Monaca Debugger. Please try to make more changes to your project and see how it runs on the debugger. Enjoy developing with Monaca!

# Part 3: Building Monaca App

In this tutorial, we only cover the building of Monaca Apps for iOS and Android:

1. [Building a Monaca App for iOS](#)
2. [Building a Monaca App for Android](#)

For more information regarding the building of Monaca Apps for other platforms, please refer to [Build](#).

## Building a Monaca App for iOS

This section will cover an instruction of how to create a Debug Build version of your app for iOS which will be installed on the development device. For more information about other types of build, please refer to [Types of Build](#).

*Prerequisite:*

- You must enroll in [Apple Developer Program](#).

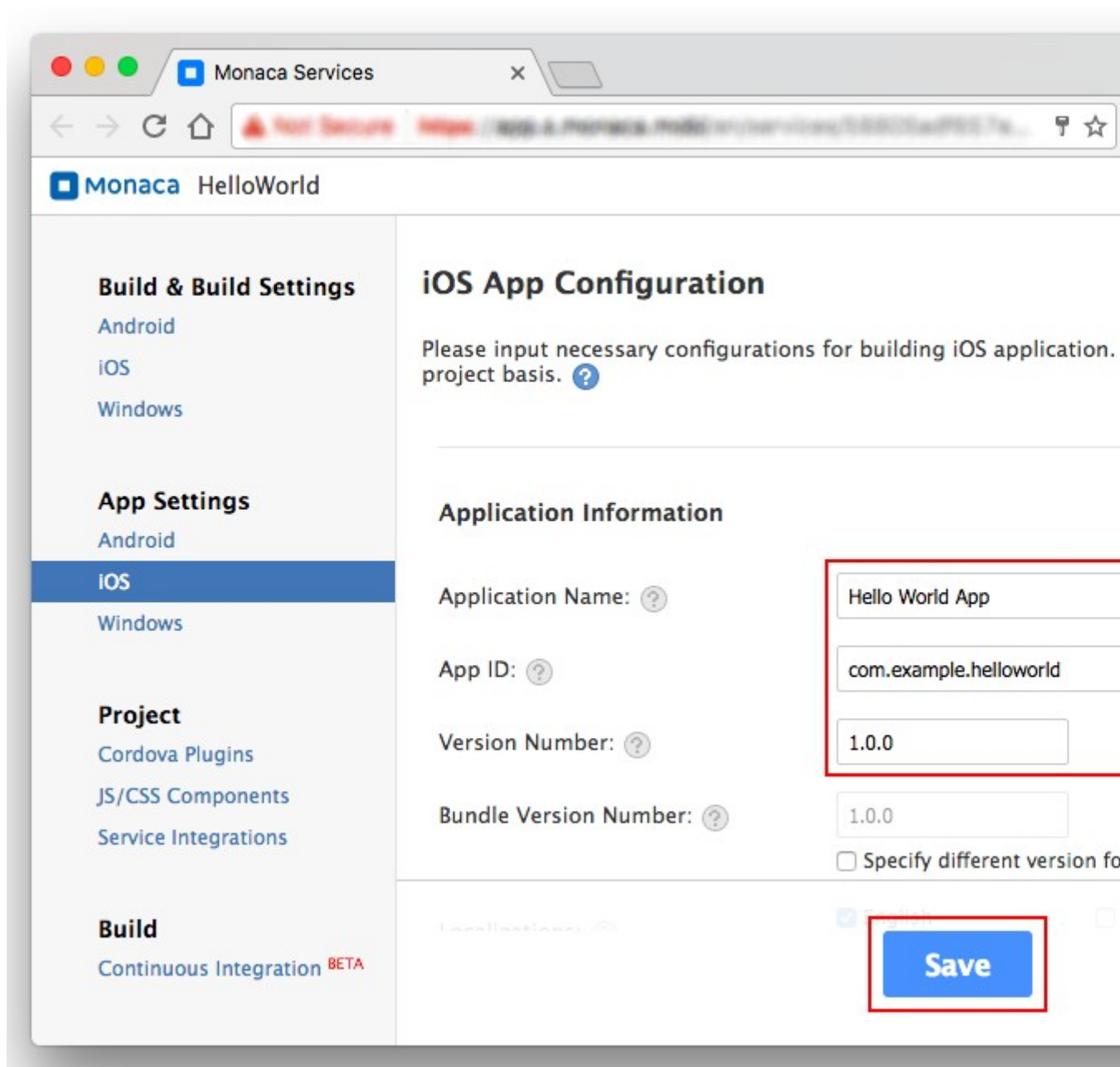
*Requirement*

You are required to create the following items from iOS Provisioning Portal after enrolling Apple Developer Program:

- App ID (see [Register App ID](#) and [Register Development Devices](#))
- Developer Certificate (see [Generate Certificates](#))
- Development Provisioning Profile (see [Create Provisioning Profiles](#))

### Step 1: Configuring iOS App Settings

1. From Monaca Cloud IDE menu, go to **Configure** → **App Settings for iOS**.
2. Fill in the necessary information of your app:
  - **Application Name**: a name representing your app publicly such as in the Market.
  - **App ID**: a unique ID representing your app. It is recommended to use reverse-domain style (for example, mobi.monaca.appname) for App ID. Only alphanumeric characters and periods (at least one period must be used) are allowed. Each segment separated by a period should begin with an alphabetic character.
  - **Version Number**: a number representing the version of your app which will be required when uploading (publishing process) your application via App Store Connect later. It needs 3 numbers separated by dots (for example, 1.10.2). Each number should be in [0-99].
  - The remaining information is optional. You can configure icon, splash screen and other configurations in the page.

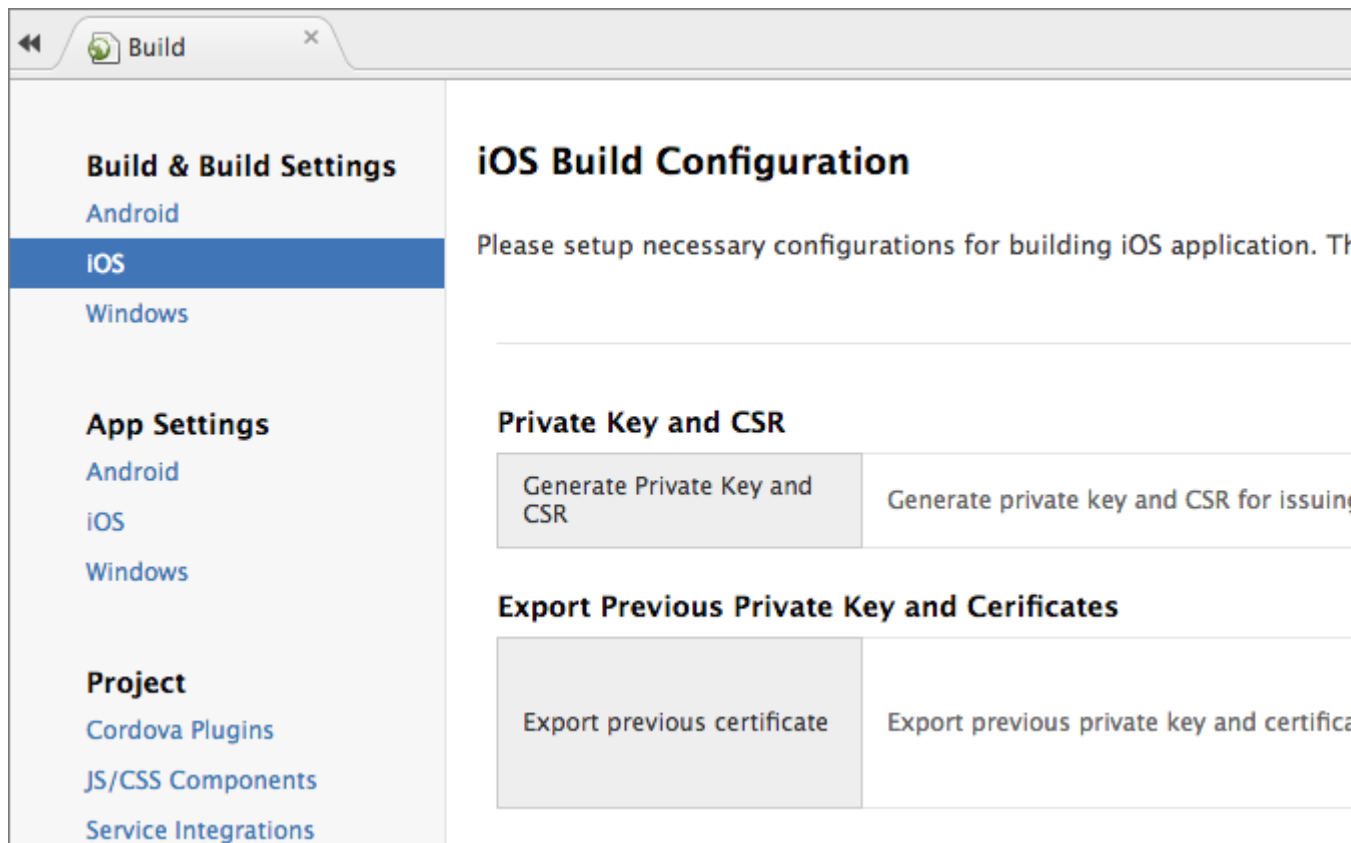


The App ID in Monaca Build Settings must be the same as the App ID you have registered in iOS Provisioning Portal. This App ID (in Monaca Build Settings) cannot contain asterisk (\*); otherwise, the build will fail.

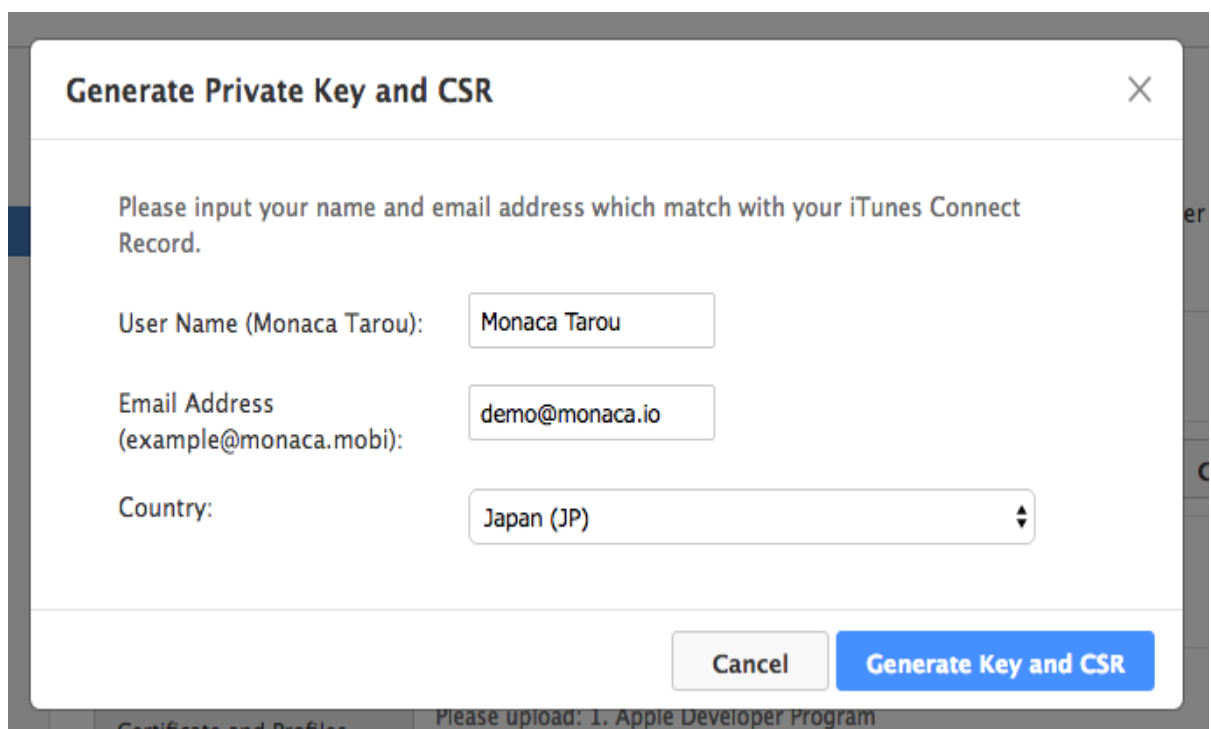
3. After finishing the configurations, click Save.

## Step 2: Configuring iOS Build Settings

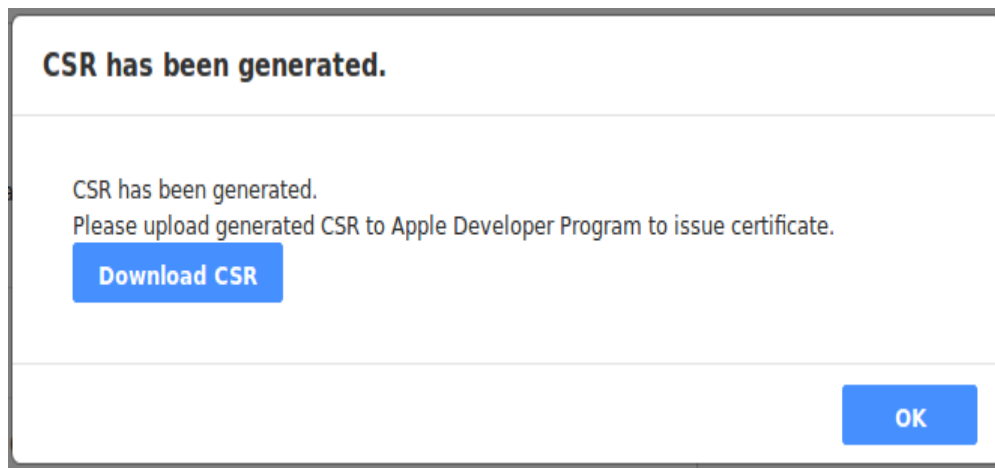
1. From Monaca Cloud IDE menu, go to **Configure** → **iOS Build Settings** .
2. You can either create a new private key or [import an existing one](#). For this tutorial, we will show how to create a new private key. Therefore, let's click on Generate Key and CSR button.



3. Fill in your Apple ID information (user name & email address) and country. Then, click on Generate Key and CSR button.



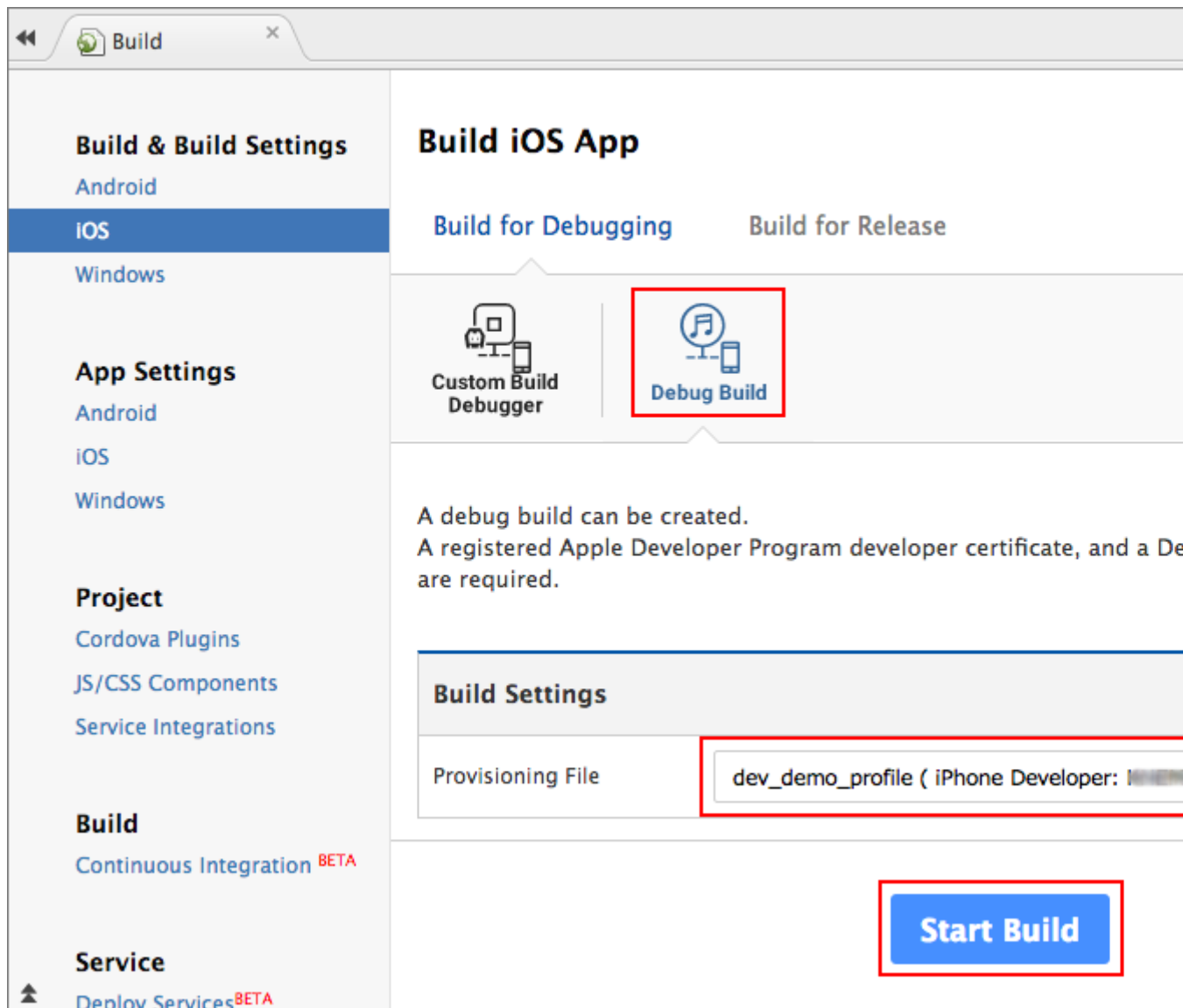
4. The following dialog box will appear if your authentication is successful. Click Download CSR button. Downloaded CSR file will be required to issue the certificates later in iOS Provisioning Portal.



If you import an existing private key, you need to use the certificates which are issued based on that imported private key. However, if you create a new private key and CRS file, you will need to use the new CRS file to issue new certificates.

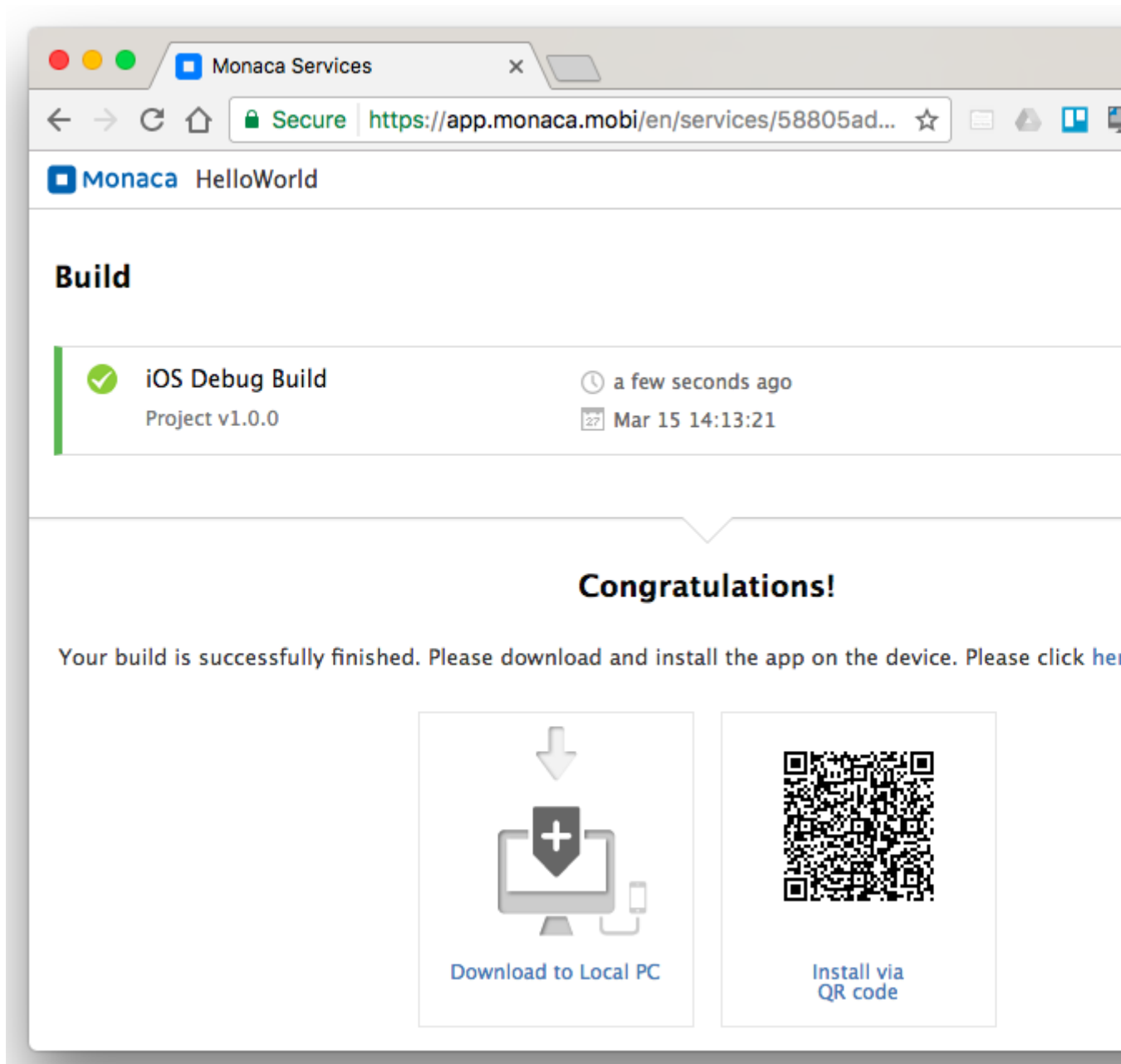
### Step 3: Building the App

1. From the Monaca Cloud IDE menu, go to **Build** → **Build App for iOS** .
2. Select the **Debug Build** option and the corresponding provisioning profile. Then, click **Start Build** button.



3. It may take several minutes for the build to complete. Please wait. The following screen will appear after the build is complete.





#### Step 4: Installing the Built App

There are 3 ways to install the debug built app such as:

1. Download the built app and use Apple Configurator 2 to install the built app on your iOS device. (Mac Only)
2. Install via QR code.
3. Install via [configured deployment services](#).

### Building a Monaca App for Android

This section will cover an instruction of how to create a Debug Build version of your app for Android. For more information about other types of build, please refer to [Types of Build](#).

## Step 1: Configuring Android App Settings

1. From Monaca Cloud IDE menu, go to **Configure** → **App Settings for Android** .
2. Fill in the necessary information of your app:
  - **Application Name**: a name representing your app publicly such as in the Market.
  - **Package Name**: a unique ID representing your app. It is recommended to use reverse-domain style (for example, mobi.monaca.appname) for App ID. Only alphanumeric characters and periods (at least one period must be used) are allowed. Each segment separated by a period should begin with an alphabetic character.
  - **Version Number**: a number representing the version of your app. It needs 3 numbers separated by dots (for example, 1.10.2). Each number should be in [0-99].
  - **Use Different Package Name for Debug Build**: if checked, the package name of the debug-built app and custom-built debugger are different. In other words, the package name of debug-built app will have `.debug` extension, and the one for project debugger will have `.debugger` 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).

- The remaining information is optional. You can configure icon, splash screen and other configurations in the page.

The screenshot shows the 'Build & Build Settings' page in the Monaca Cloud IDE. The left sidebar contains a navigation menu with sections: 'Build & Build Settings' (containing 'Android', 'iOS', 'Windows'), 'App Settings' (containing 'Android', 'iOS', 'Windows'), 'Project' (containing 'Cordova Plugins', 'JS/CSS Components', 'Service Integrations'), 'Build' (containing 'Continuous Integration BETA'), and 'Service'. The 'Android' option under 'App Settings' is selected. The main content area is titled 'Android App Configuration' and contains the 'Application Information' section. This section includes input fields for 'Application Name' (filled with 'Hello World App'), 'Package Name' (filled with 'monaca.demo.helloworld'), 'Version' (filled with '1.0.0'), and 'Version Code'. There are checkboxes for 'Use Different Package Name for Debug Build' (disabled), 'Specify the version code' (disabled), and 'Fullscreen' (disabled). A blue 'Save' button is located at the bottom right of the configuration area. Red boxes highlight the 'Application Name', 'Package Name', 'Version' fields and the 'Save' button.

3. After finishing the configurations, click Save.

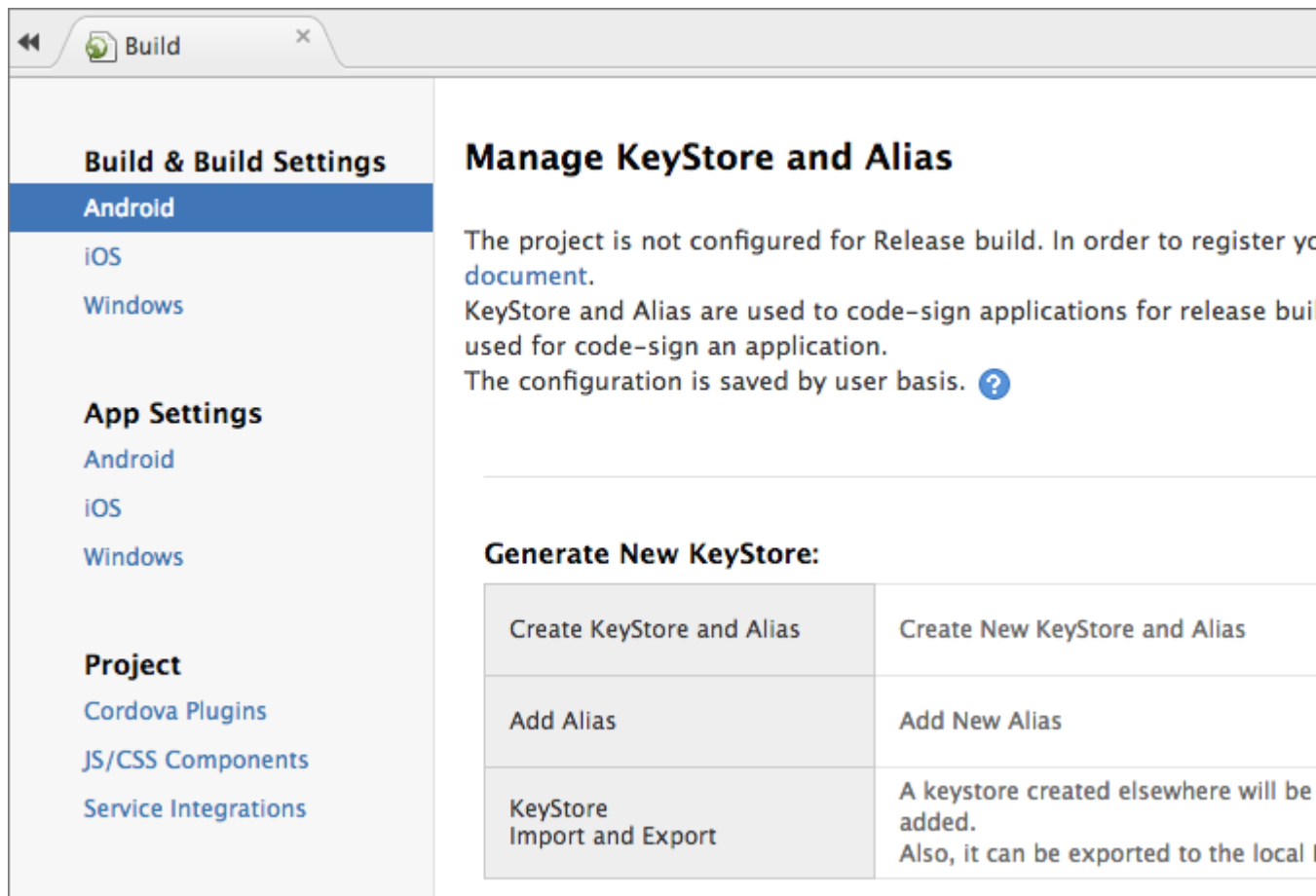
## Step 2: Configuring Android KeyStore

Android KeyStore is used for storing the keys (Alias) needed to sign a package. When a KeyStore is lost or it is overwritten by another KeyStore, it is impossible to re-sign the signed package with the same key. One KeyStore can contain multiple Alias, but only one Alias is used for code-sign an application.

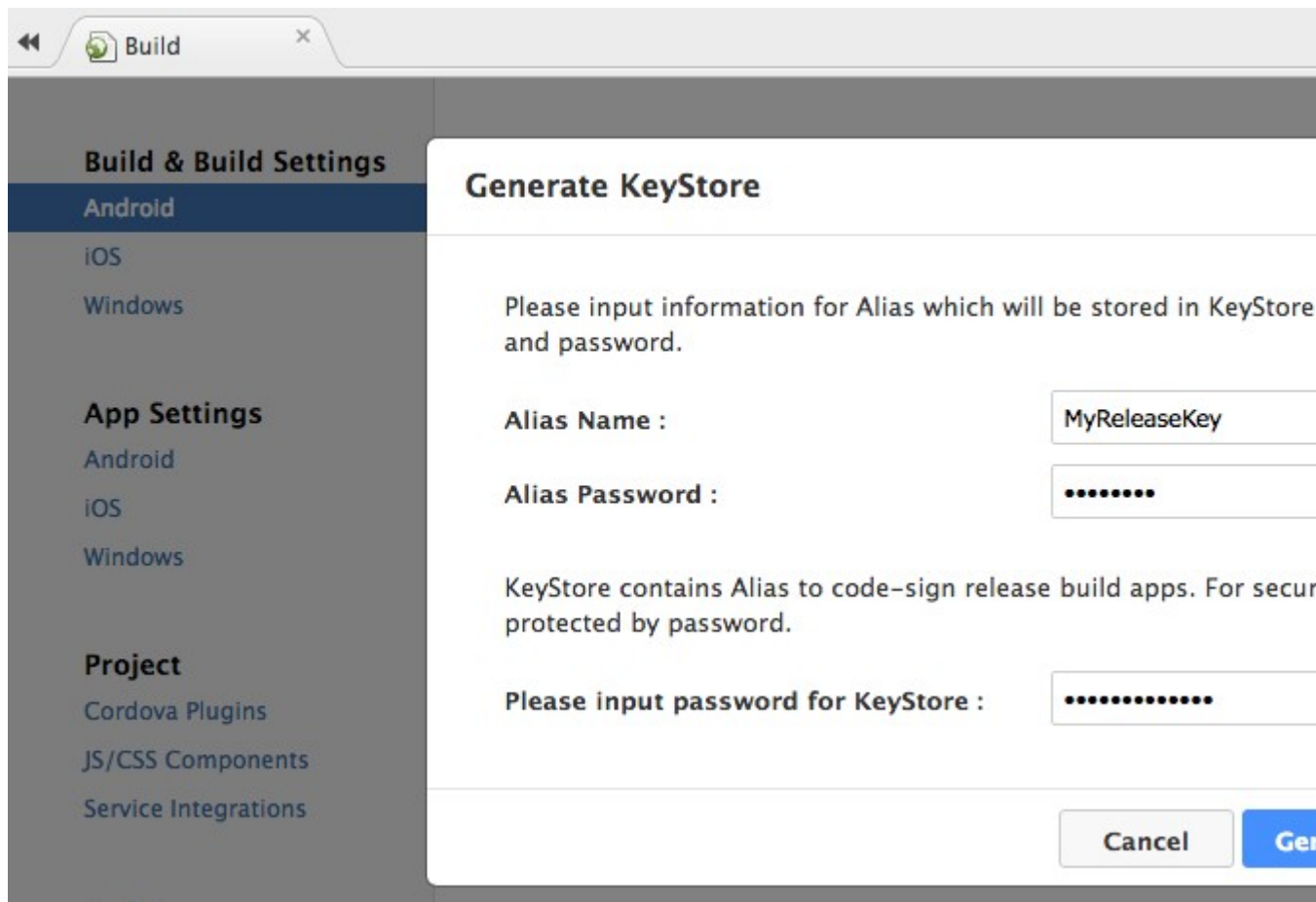
In order to configure Android KeyStore in Monaca, please do as follows:

1. From Monaca Cloud IDE menu, go to **Configure** → **Android KeyStore Settings**.
2. You will need to generate a new KeyStore if you haven't created one yet. The KeyStore can either be created or imported. In this tutorial, we assume that you need to create a new

KeyStore. Therefore, click on Clear and Generate New button. Then, the following screen will appear:



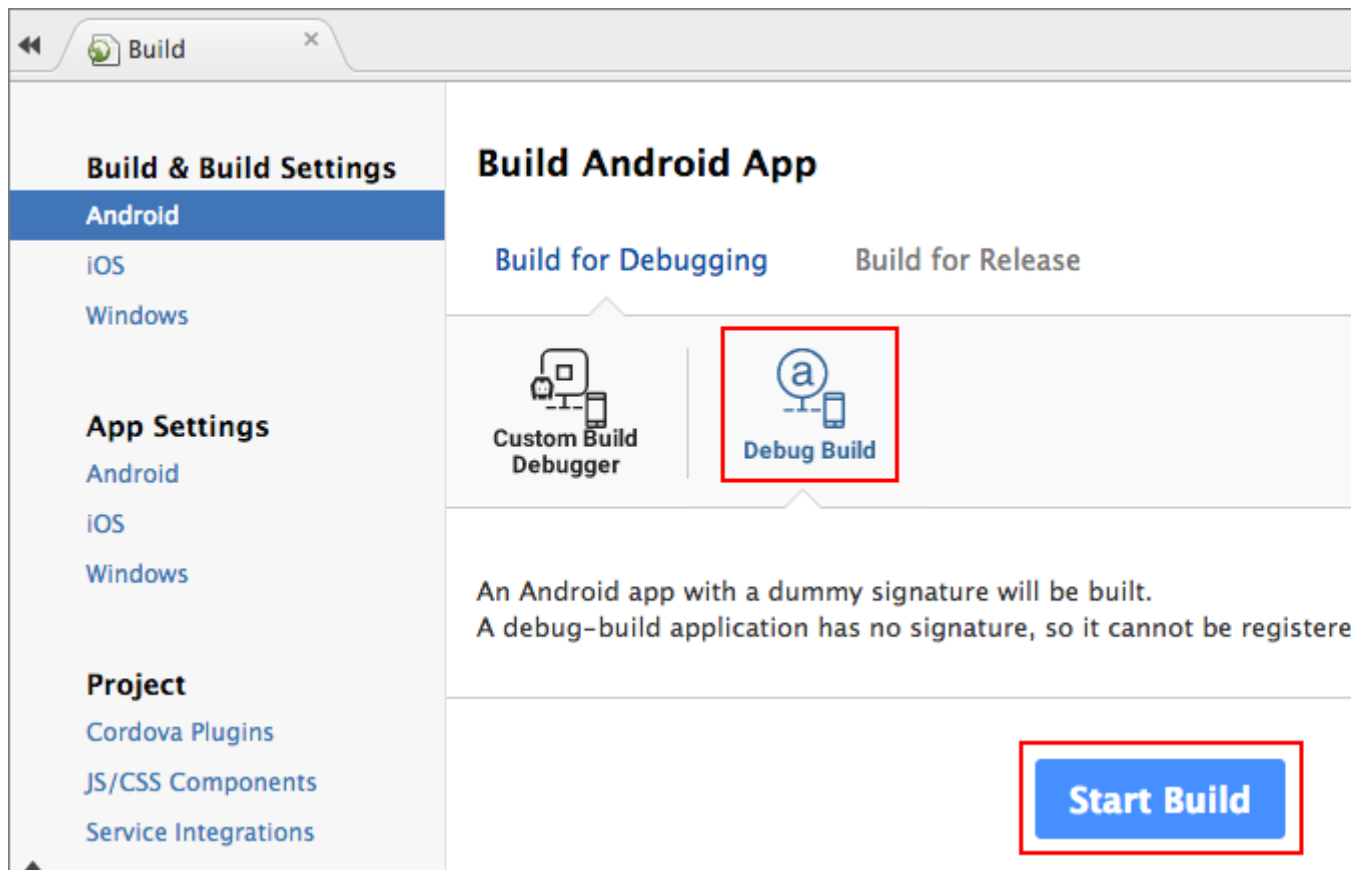
- Fill in the necessary information related to the Keystore such as:
  - Alias Name:** key information stored in the Keystore which is used to sign an app package.
  - Alias Password:** password for the Alias.
  - Keystore Password:** password for the new Keystore.



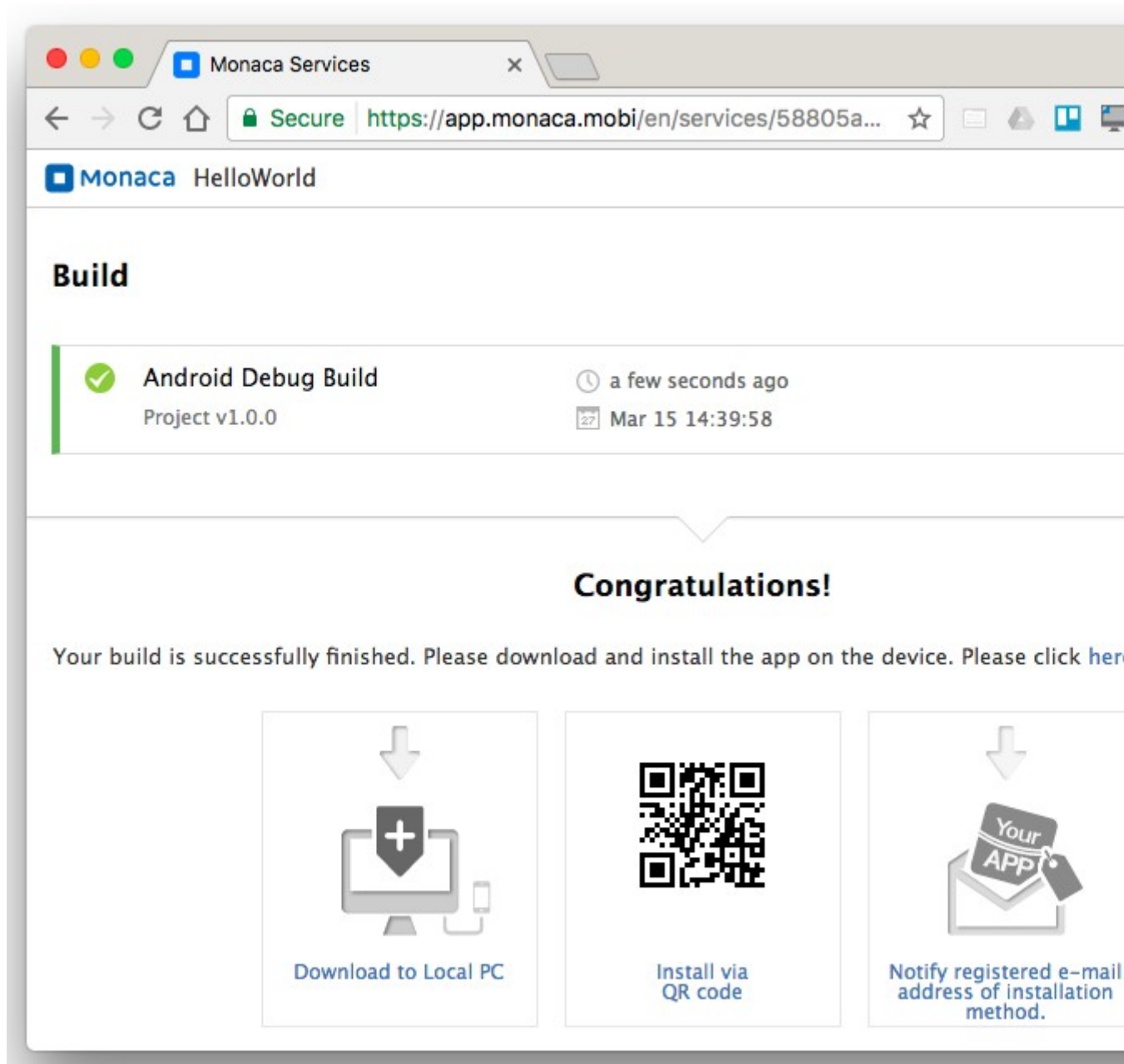
4. Then, click on Generate KeyStore and Alias button.

### Step 3: Building the App

1. From the Monaca Cloud IDE menu, go to **Build** → **Build App for Android** .
2. Select Debug Build option and click on Start Build button.



3. It may take several minutes for the build to complete. Please wait. The following screen will appear after the build is successfully completed.



## Step 4: Installing the Built App

There are 5 ways you can install the built app such as:

1. using [Network Install](#)
2. installing via QR Barcode
3. downloading the built app directly to your computer and installing it via USB cable
4. sending the URL to download the built app to your registered email address
5. installing via configured [deployment services](#).

Next:

- [Part 4: Publishing Monaca App](#)

# **Part 4: Publishing Monaca App**

## **Publishing for App Store**

Please refer to [App Store Distribution](#).

## **Publishing for Google Play**

Please refer to [Google Play Distribution](#).

For more information regarding the distribution of Monaca Apps for other platforms, please refer to [Distribution](#).



# Google Play Distribution

## Prerequisite

In order to distribute your apps through Google Play, you must register as a *Google Play Developer Console* account. The registration fee is \$25 (one-time only). It will take 48 hours for your *Google Play Developer Console* registration to be fully processed. To register, please go [here](#).

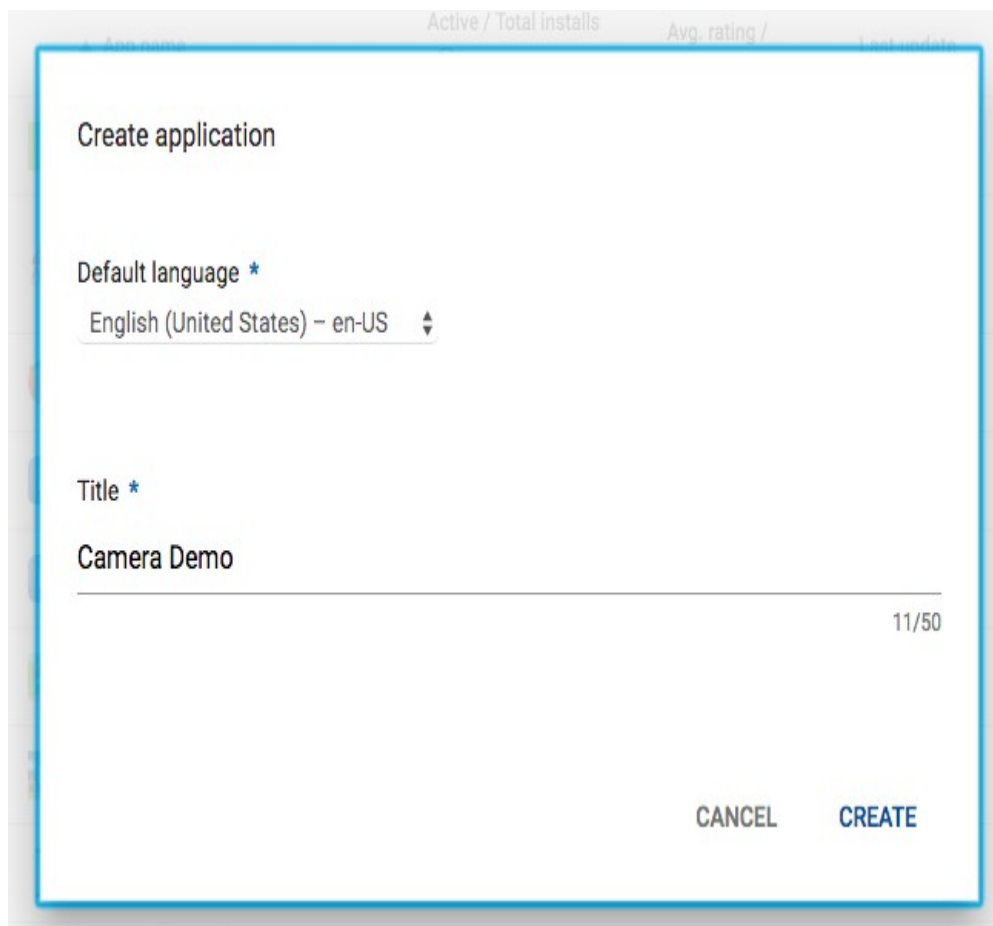
If you want to sell your apps, you will also need to register as a Google Checkout merchant as well. To register, please visit [Link a Google Play Developer account to your payments profile](#).

## Create a Release Build of the App

Create and download a release build version of your app through Monaca. Please refer to [Building for Android](#) to build a release build of the app. Then, download the built app (APK file).

## Register the Apps in Google Play

1. Go to [Google Play Developer Console](#) and login with a valid Google Developer account.
2. Click on CREATE APPLICATION button.
3. Choose a default language and enter a title for your app. Then, click CREATE. Then, you will be forwarded to Store listing page.



The screenshot shows a 'Create application' dialog box with a blue border. At the top, there is a header bar with columns: 'App name', 'Active / Total installs', 'Avg. rating /', and 'Last update'. The dialog contains the following fields:

- Create application** (Section header)
- Default language \***: A dropdown menu showing 'English (United States) - en-US'.
- Title \***: A text input field containing 'Camera Demo'.
- A character count '11/50' is displayed at the bottom right of the title field.
- At the bottom right, there are two buttons: 'CANCEL' and 'CREATE'.

4. In this page, you will need to fill in the following necessary information and click **SAVE DRAFT**.

<b>Data</b>	<b>Description</b>
Short description	Description of your app shown in Google Play. It can be up to 80 characters.
Full description	Description of your app shown in Google Play. It can be up to 4000 characters.
Screenshots	At least 2 screenshots are required but you can upload up to 8 screenshots per type.
Hi-res icon	You are required to upload a high-resolution icon (512x512 PNG file) for your app.
Feature Graphic	You are required to upload a feature graphic (1024x500 PNG file) for your app.
Application type	It can be <b>Applications</b> or <b>Games</b> . They are the major application types in Google Play.
Category	Select a category for your app.
Content rating	Select a content rating of your app as appropriate.
Contact details	You must have at least one support channel for your app. The support channels can be website, email and phone. This information can be viewed by users from Google Play.
Privacy Policy	Input your privacy policy URL if you have one. Otherwise, please tick <b>NOT submitting a privacy policy URL at this time..</b>

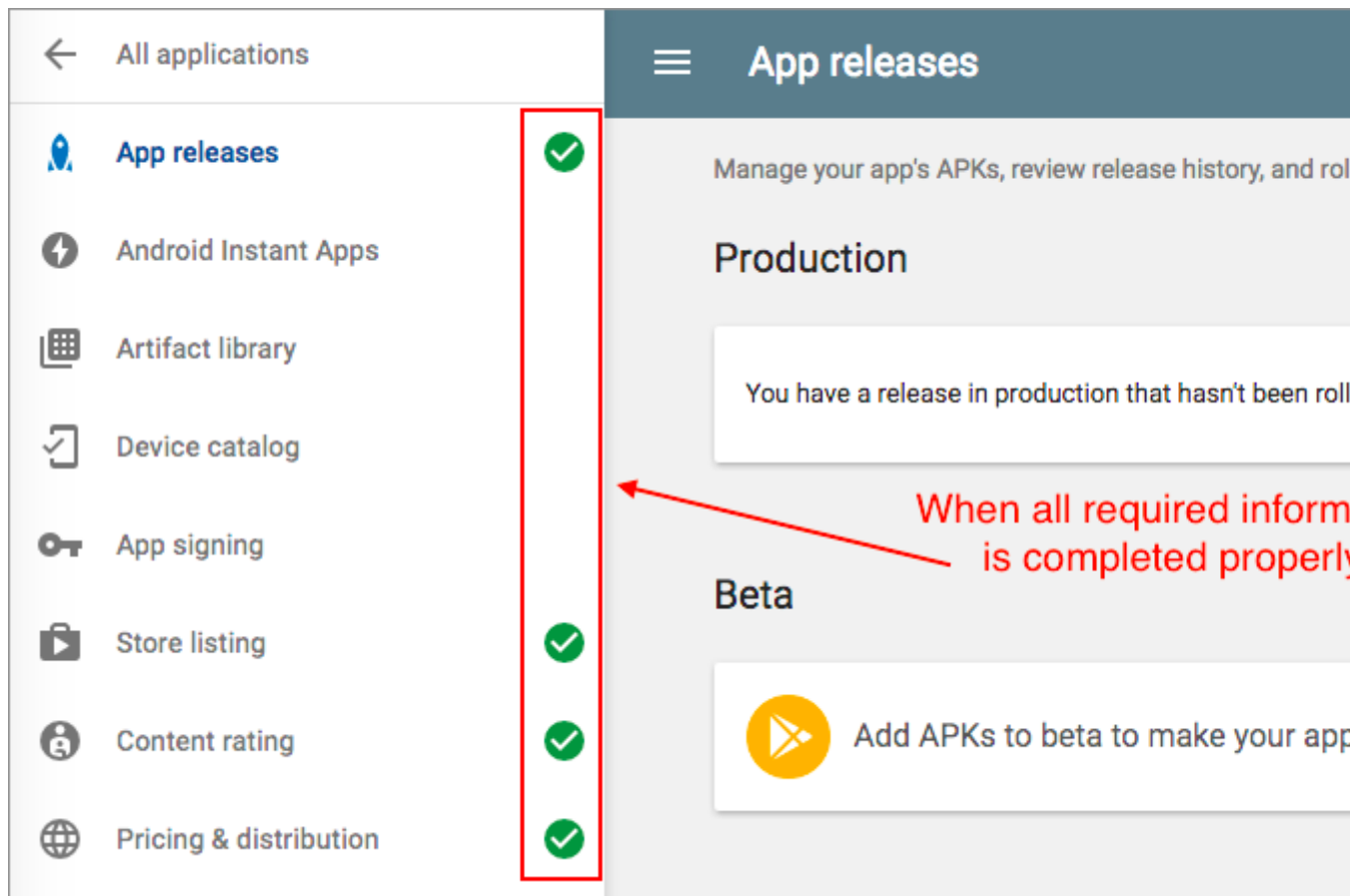
5. Go to **Content rating** section and click **CONTINUE**.
6. Fill in your email address and select your app category. Then, you will be asked to answer some questions related to the selected app category.
7. Click on **SAVE QUESTIONNAIRE** and **CALCULATE RATING**.
8. Confirm your rating information and click **APPLY RATING**.
9. Go to **Pricing & distribution** section. In this page, you will be asked to complete various questions related the price and availability of your app. Until this point, you are successfully complete the necessary information for your app to be ready for a release.

## **Release the App**

In this section, we will start uploading the APK file and finalize the release information of the app before submitting to the Play Store.

1. Go to **App releases** section. In this page, you can upload your APK files for testing (Beta and Alpha) and production.
2. Under the **Production** section, click on **MANAGE PRODUCTION**. Then, click on **CREATE RELEASE**.
3. Then, you will need to:
  - complete the google Play App Signing option
  - upload the APK file

- configure the Release name (release version)
  - write about what's new about this release
4. Click SAVE and REVIEW.
  5. Review your app's release information one more time and click on START ROLLOUT TO PRODUCTION button to submit your app to the Play Store. Please note that you can't submit your app if you don't configure your app properly. In other words, the START ROLLOUT TO PRODUCTION button is disabled unless the required information is completed properly.



See Also:

- [Building for Android](#)
- [App Store Distribution](#)
- [Non-market App Distribution](#)

# 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"><li>• QR Code</li><li>• <a href="#">Network Install</a></li><li>• Sideload</li></ul>
Release Build	A signed package with the developer's code sign which can be distributed in the market	<ul style="list-style-type: none"><li>• Sideload</li><li>• Google Play Store and other eligible markets</li></ul>

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.

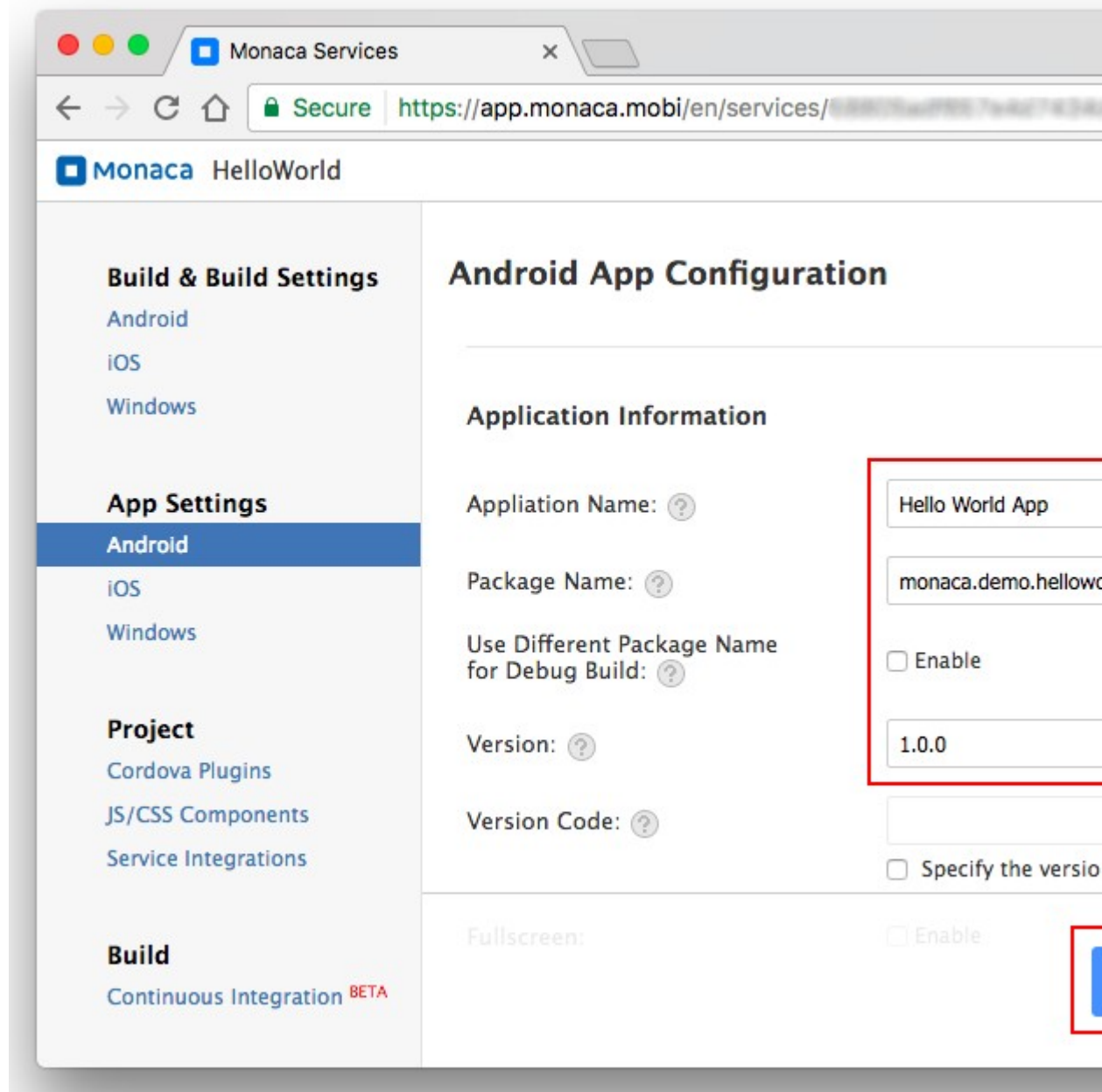
## Step 1: Configure Android App

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. 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).
Use Different Package Name for Debug Build	
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.

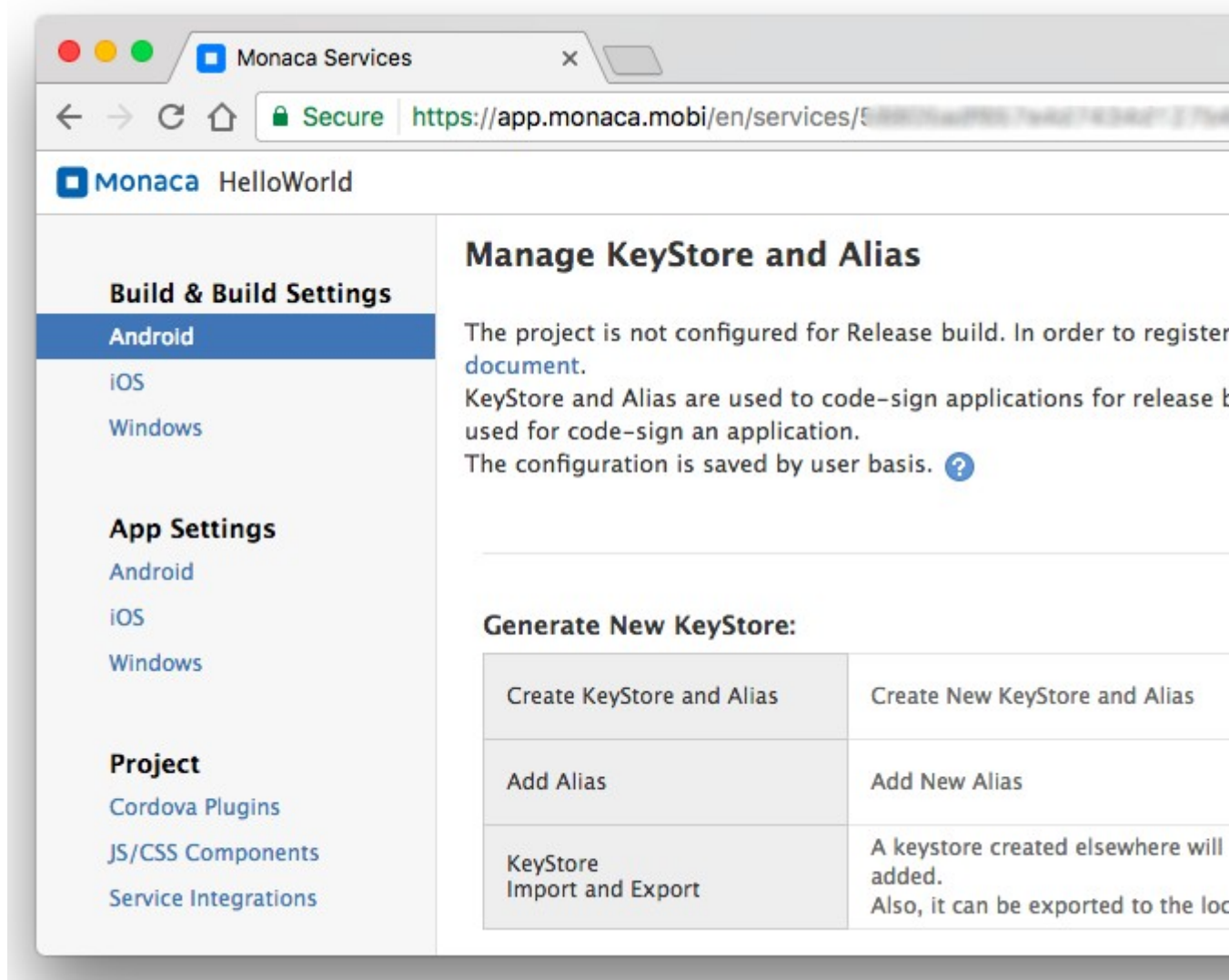
```
<platform name="android">
  <icon src="/res/android/icon/ldpi.png" density="ldpi"
background="/res/android/icon/ldpi-background.png"
foreground="/res/android/icon/ldpi-foreground.png"/>
  <icon src="/res/android/icon/mdpi.png" density="mdpi"
background="/res/android/icon/mdpi-background.png"
foreground="/res/android/icon/mdpi-foreground.png"/>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

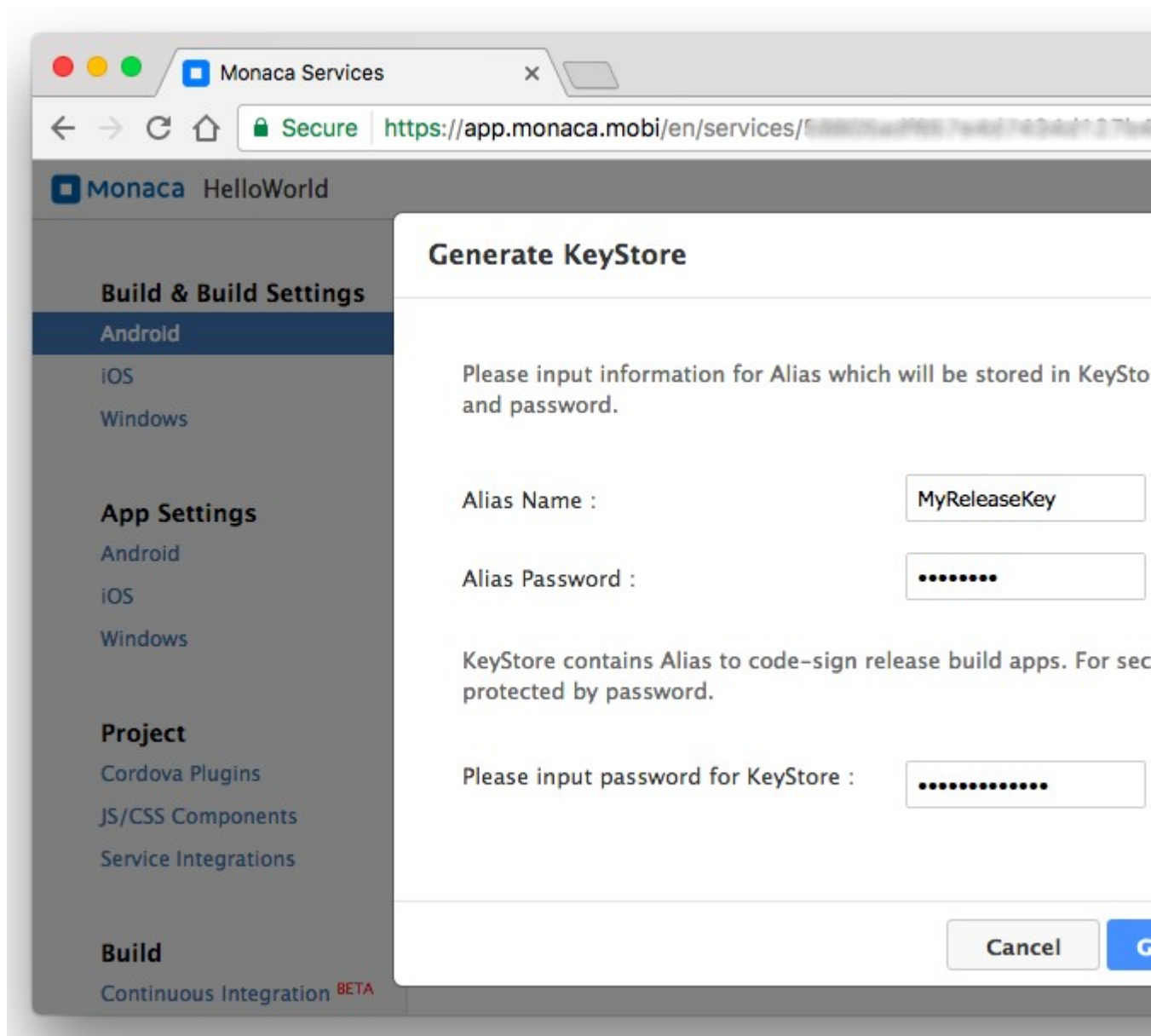
A keystore is required for the building of a release version for your Android app. In Monaca, you can either create a new keystore or import an existing one. In order to create a new keystore, please do as follows:

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.

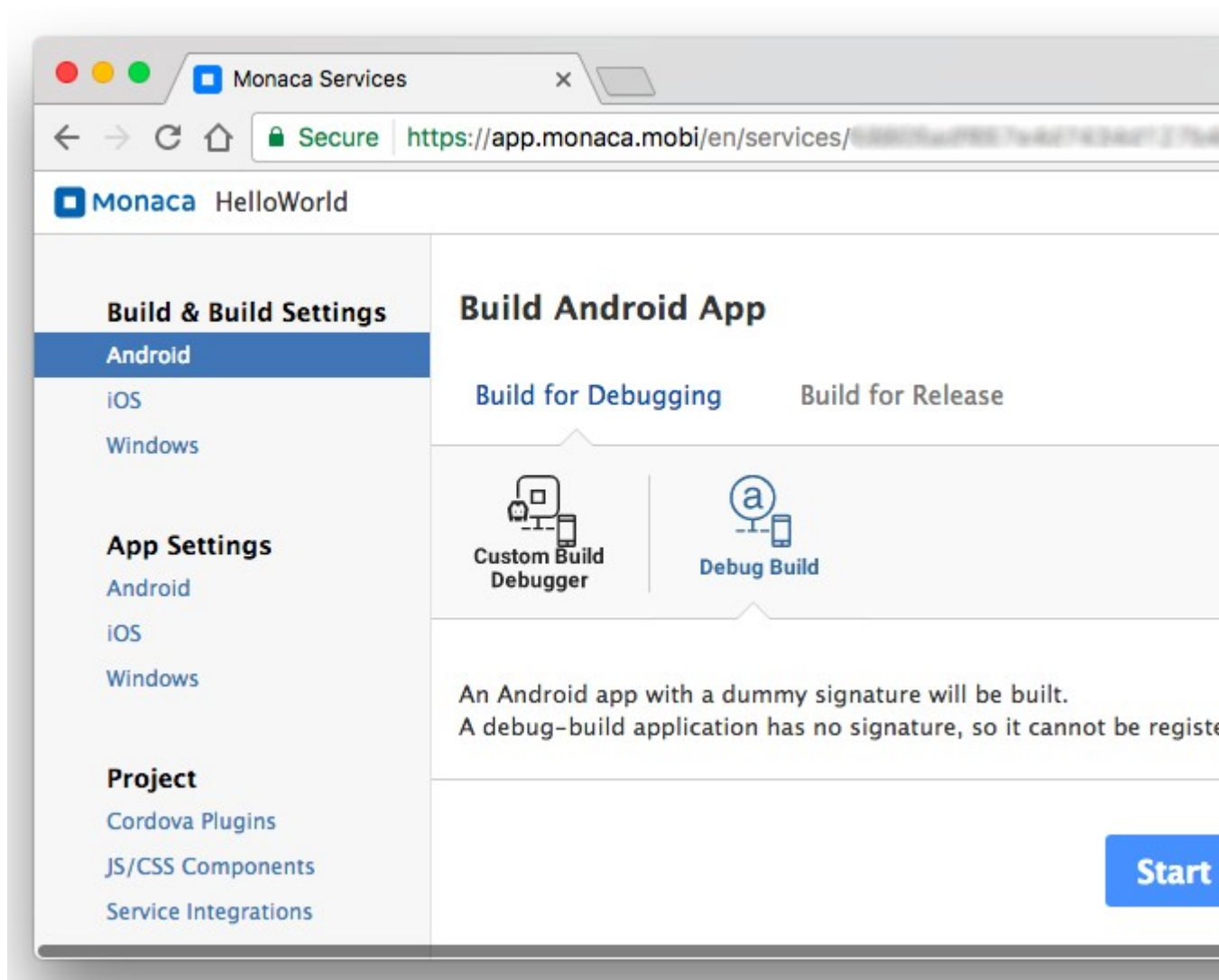
5. Then, click on Generate Keystore and Alias button to Generate the keystore.

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.

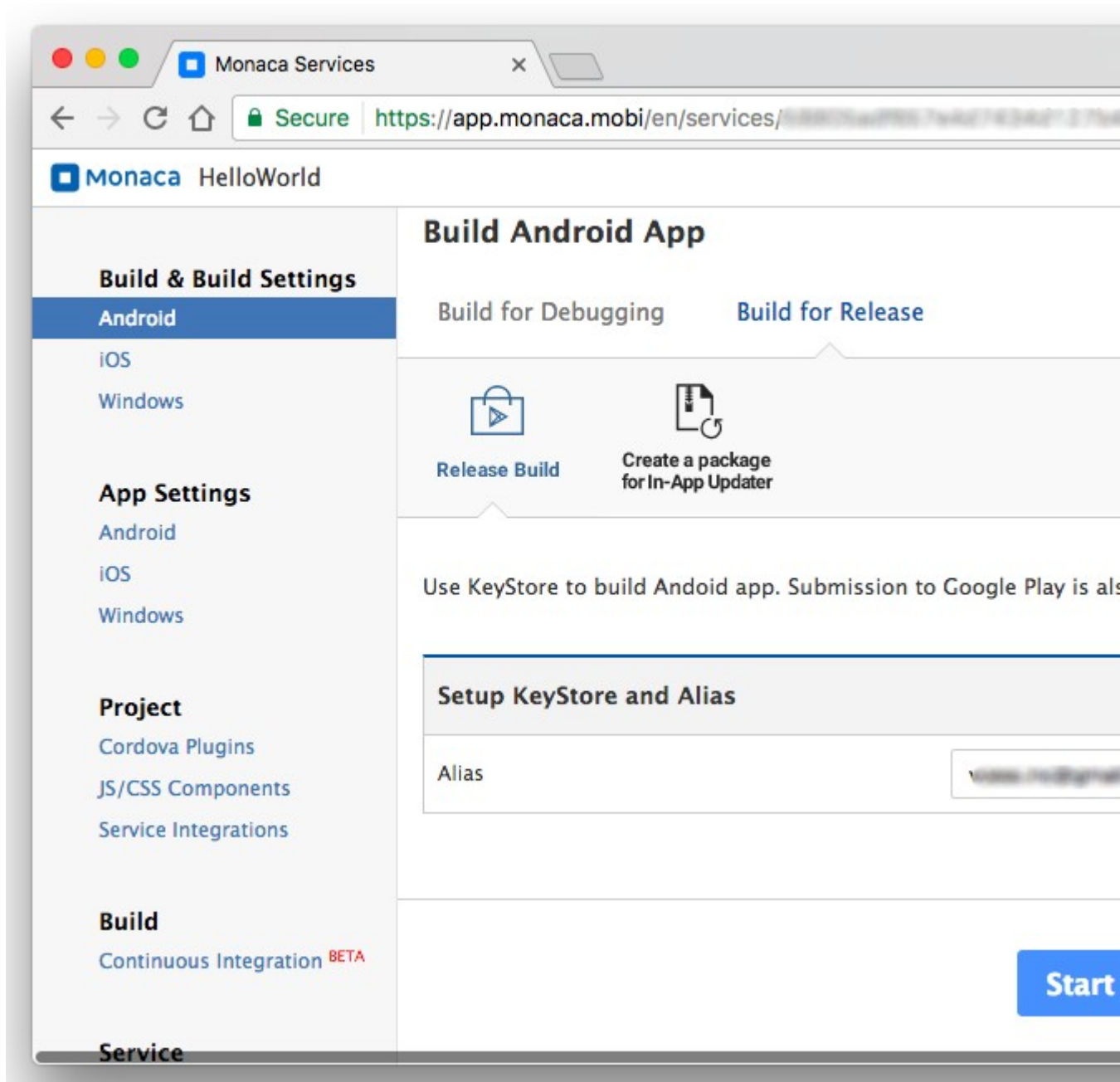
## Step 3: Start Building

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**.

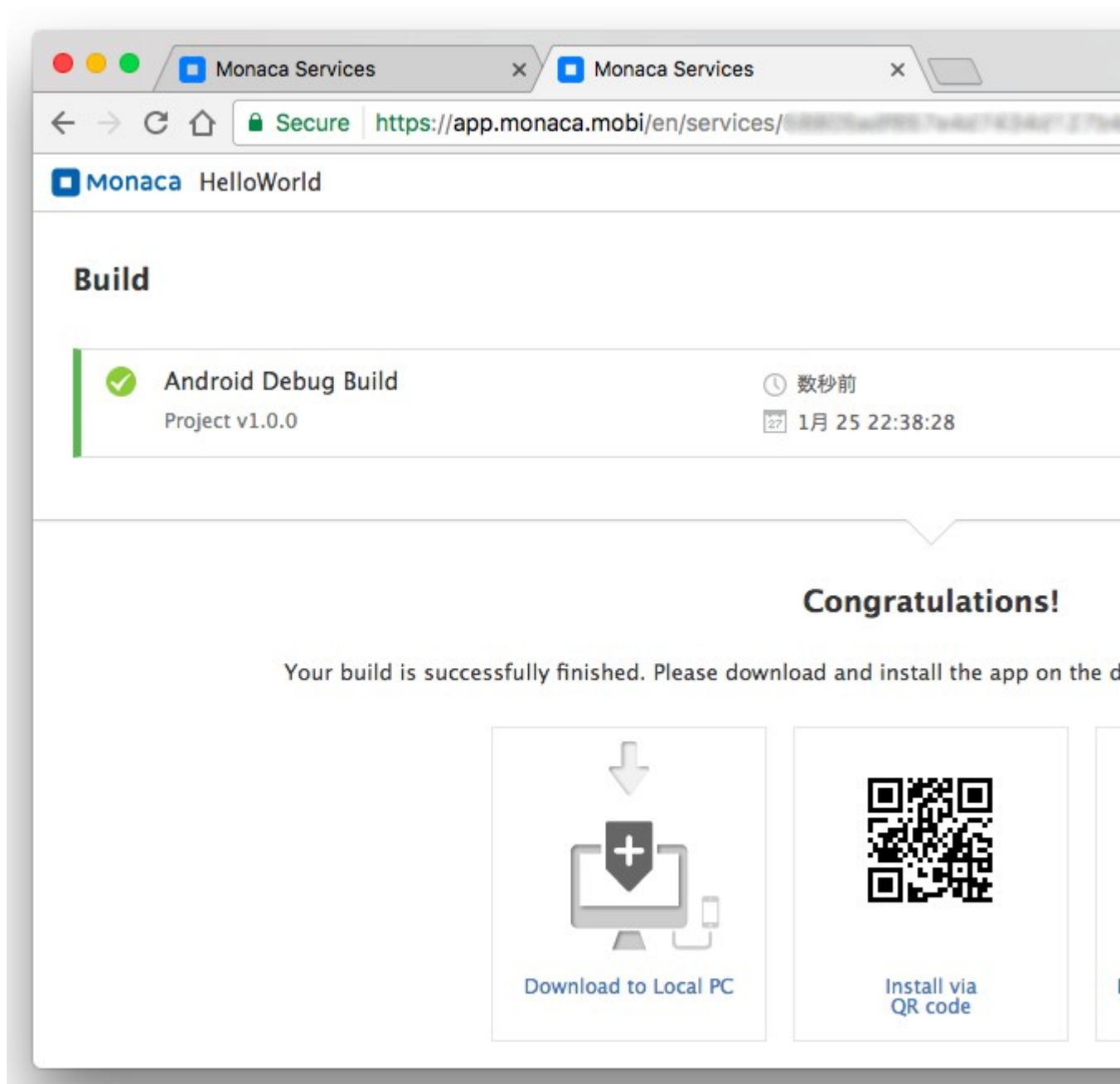




3. If you choose **Release Build**, you will also need to select an alias to sign your package before start building.



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](#)

[https://docs.monaca.io/en/products\\_guide/monaca\\_ide/tutorial](https://docs.monaca.io/en/products_guide/monaca_ide/tutorial)

# Integrated Terminal

Monaca Cloud IDE is equipped with terminal consoles. This is mainly used for running a HTTP server for showing preview application (Previewer). It can also be used to provide access to the Linux container for your project.

In this section, we will guide you through how you can take advantages of the Integrated Terminal feature in Monaca Cloud IDE.

Terminal feature can be used from pay plan.

## Overview

Integrated Terminal is equipped with two console windows:

1. **Preview Log** tab: a terminal window for running HTTP service for providing content in Previewer
2. **Terminal** tab: a terminal window for executing arbitrary commands (i.e. git, npm, cordova, monaca and most of the UNIX basic commands)

When the user starts Cloud IDE, a Linux container is created on demand. Containers are shared among multiple users, which means you cannot execute certain commands or operations that are restricted.

Container is destroyed soon after user closes Cloud IDE. Be sure to read the following section to prevent important files to be deleted.

## Storage & Directory

All files that are created in the container will be lost when the container is destroyed, except of the files located in the following directories:

- `/home/terminaluser`: user's home directory. This directory is shared among all projects
- `/project`: project root directory

## Using Integrated Terminal

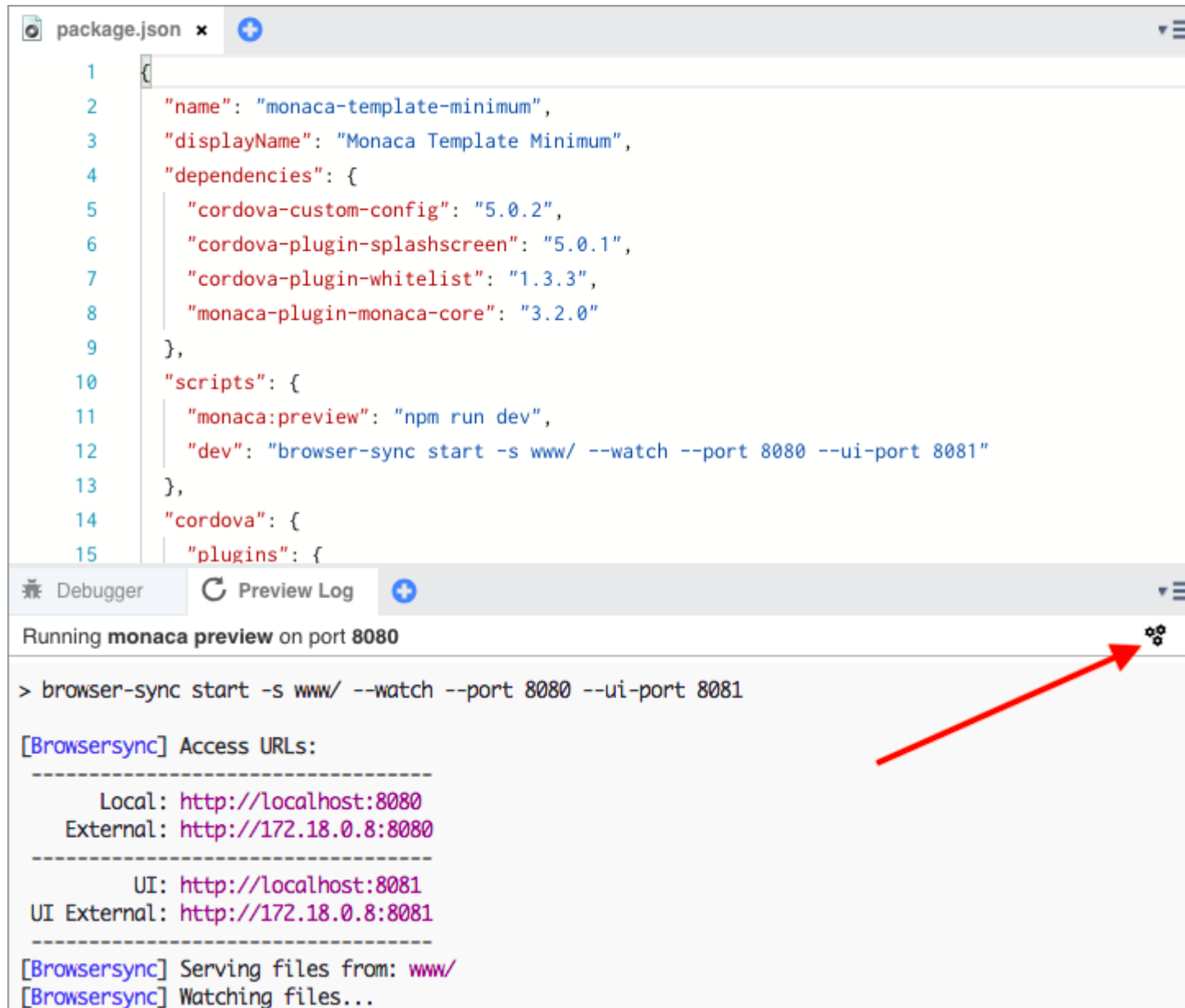
### Starting Up

When the project is opened in Monaca Cloud IDE for the first time, it will perform the necessary configuration for the project under certain circumstances:

- When the project is equipped with `package.json`, it will execute `npm install` to install required NPM dependencies.
- When the `monaca:preview` script is missing from the `package.json` file, the update dialog is displayed and it will execute `monaca update` to update the project structure.

## Preview Log and Preview Server

Preview Log tab shows the output from the Preview server for each project. The Preview Server runs `monaca preview` command which executes `monaca:preview` script of the `package.json` file. As you can see in this picture that the preview server is running `browser-sync` as it is defined in the script. Previewer will connect to the Preview server once it is accepting HTTP request. The default port number is 8080 but you can change it by clicking the gear icon.



The screenshot displays the Monaca IDE interface. The top editor shows the `package.json` file with the following content:

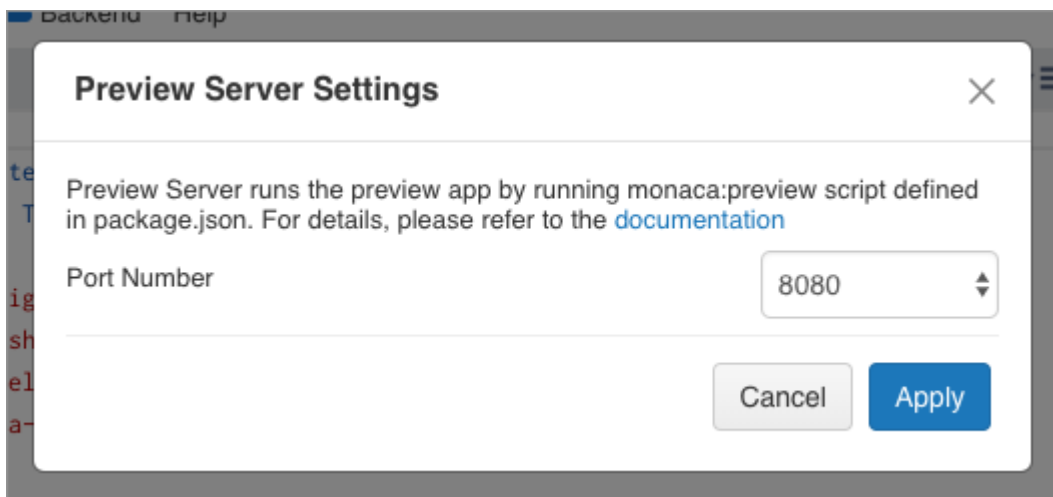
```
1 {
2   "name": "monaca-template-minimum",
3   "displayName": "Monaca Template Minimum",
4   "dependencies": {
5     "cordova-custom-config": "5.0.2",
6     "cordova-plugin-splashscreen": "5.0.1",
7     "cordova-plugin-whitelist": "1.3.3",
8     "monaca-plugin-monaca-core": "3.2.0"
9   },
10  "scripts": {
11    "monaca:preview": "npm run dev",
12    "dev": "browser-sync start -s www/ --watch --port 8080 --ui-port 8081"
13  },
14  "cordova": {
15    "plugins": {
```

Below the editor, the **Preview Log** tab is active, showing the output of the `monaca preview` command. The log indicates that the preview is running on port 8080. A red arrow points to a gear icon in the top right corner of the Preview Log panel, which is used to access the configuration dialog for changing the port number.

```
> browser-sync start -s www/ --watch --port 8080 --ui-port 8081

[Browsersync] Access URLs:
  -----
    Local: http://localhost:8080
    External: http://172.18.0.8:8080
  -----
    UI: http://localhost:8081
    UI External: http://172.18.0.8:8081
  -----
[Browsersync] Serving files from: www/
[Browsersync] Watching files...
```

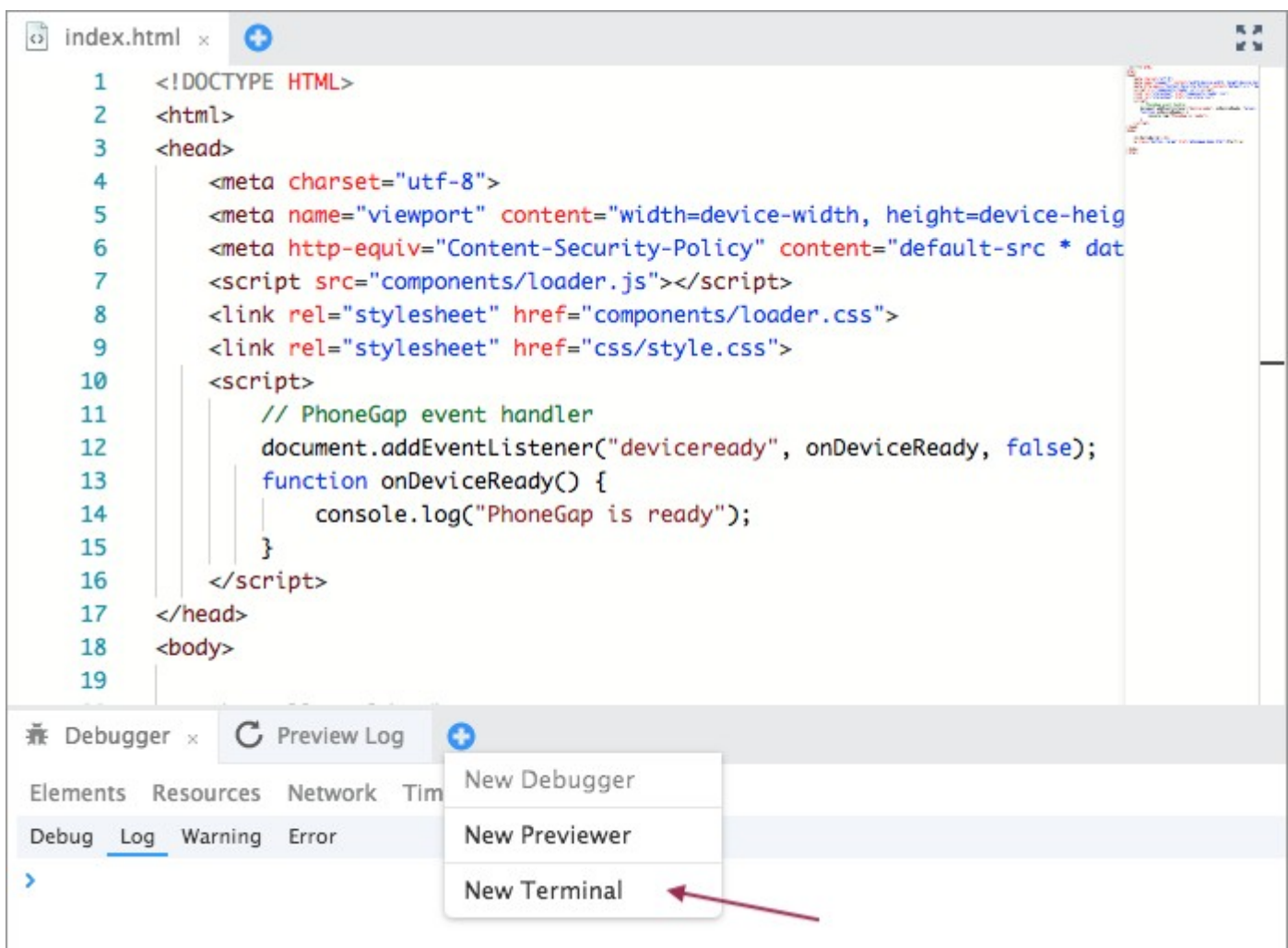
The port number is available between 8080-8084 in the configuration dialog.



When you change the port number in the Preview Server Settings dialog, you might as well need to change the port number in the package.json file and other config files manually.

## Open New Terminal

To open a terminal, you can click the plus sign next to your tabs then choose New Terminal.

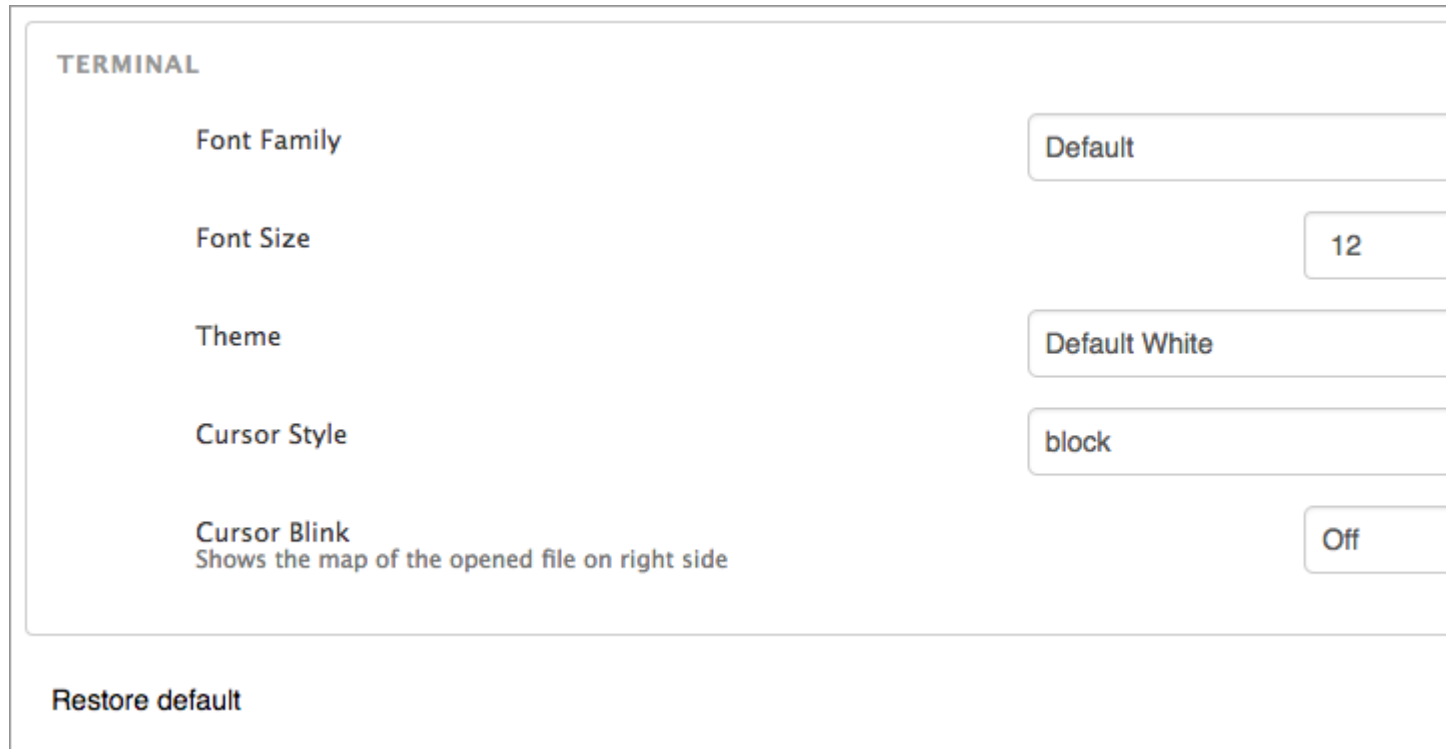


This feature is not available for free-plan users.

## Customizing the Terminal

Terminal is actually hosted by the terminal multiplexer called [Tmux](#). You can override the default Tmux configuration by editing `~/ .tmuxrc`.

Visual aspects of the terminal can be customized by going to **Configure** → **Workspace Configuration** .



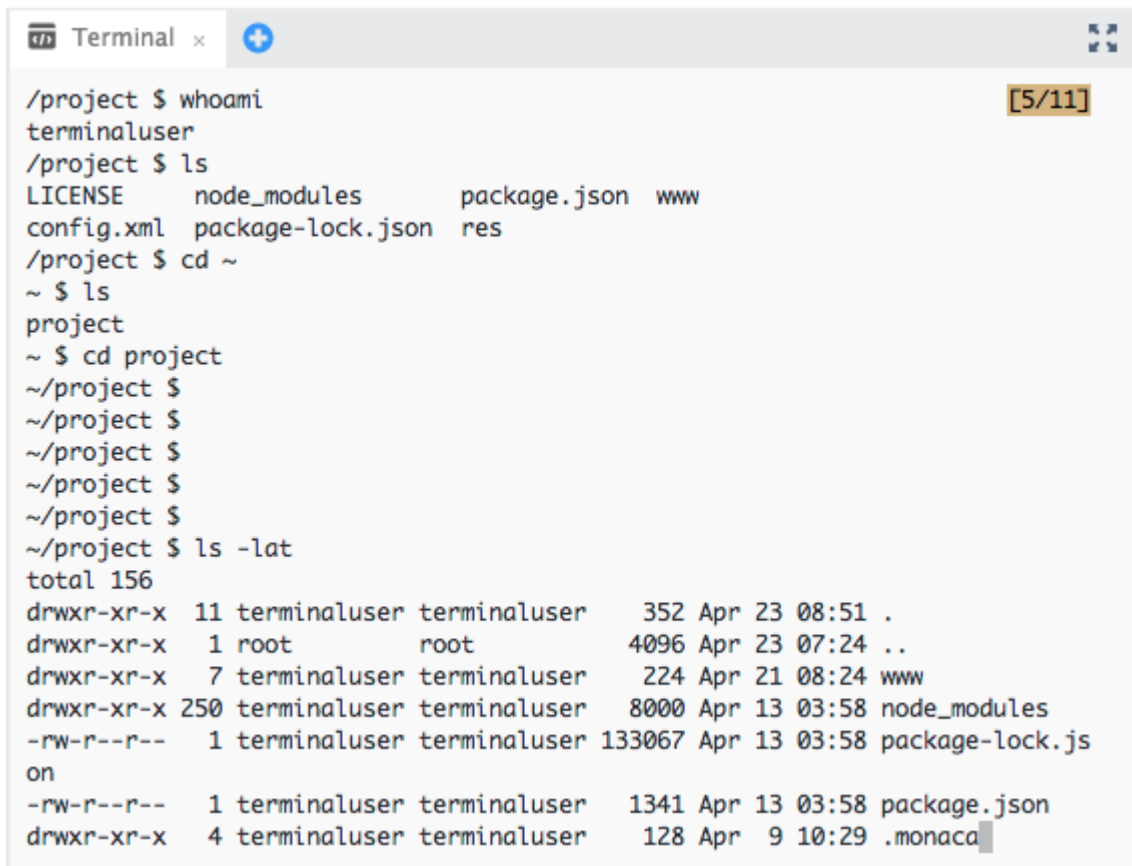
The screenshot shows a configuration window titled "TERMINAL" with a light gray background. It contains five settings, each with a label on the left and a value in a rounded rectangular box on the right:

- Font Family**: Default
- Font Size**: 12
- Theme**: Default White
- Cursor Style**: block
- Cursor Blink**: Off

Below the "Cursor Blink" setting, there is a smaller text label: "Shows the map of the opened file on right side". At the bottom left of the configuration panel, there is a button labeled "Restore default".

## Scroll mode

You might notice that once you are scrolling inside the terminal console, a little message appears at the top-right corner telling you the current scrolling position of the pane. It indicates that you are in the `scroll mode`. When you scroll down to the bottom of the terminal pane, the `scroll mode` will be exited automatically. Otherwise, you can press `Ctrl-C` or `ESC` key to leave from the `scroll mode`.



```
Terminal x +
/project $ whoami
terminaluser
/project $ ls
LICENSE      node_modules  package.json  www
config.xml   package-lock.json  res
/project $ cd ~
~ $ ls
project
~ $ cd project
~/project $
~/project $
~/project $
~/project $
~/project $
~/project $ ls -lat
total 156
drwxr-xr-x  11 terminaluser terminaluser   352 Apr 23 08:51 .
drwxr-xr-x   1 root         root         4096 Apr 23 07:24 ..
drwxr-xr-x   7 terminaluser terminaluser   224 Apr 21 08:24 www
drwxr-xr-x 250 terminaluser terminaluser  8000 Apr 13 03:58 node_modules
-rw-r--r--   1 terminaluser terminaluser 133067 Apr 13 03:58 package-lock.js
on
-rw-r--r--   1 terminaluser terminaluser  1341 Apr 13 03:58 package.json
drwxr-xr-x   4 terminaluser terminaluser   128 Apr  9 10:29 .monaca
```

## FAQ/Troubleshooting

There are several known issues & limitations.

### Browser with Ad-Blocker Settings

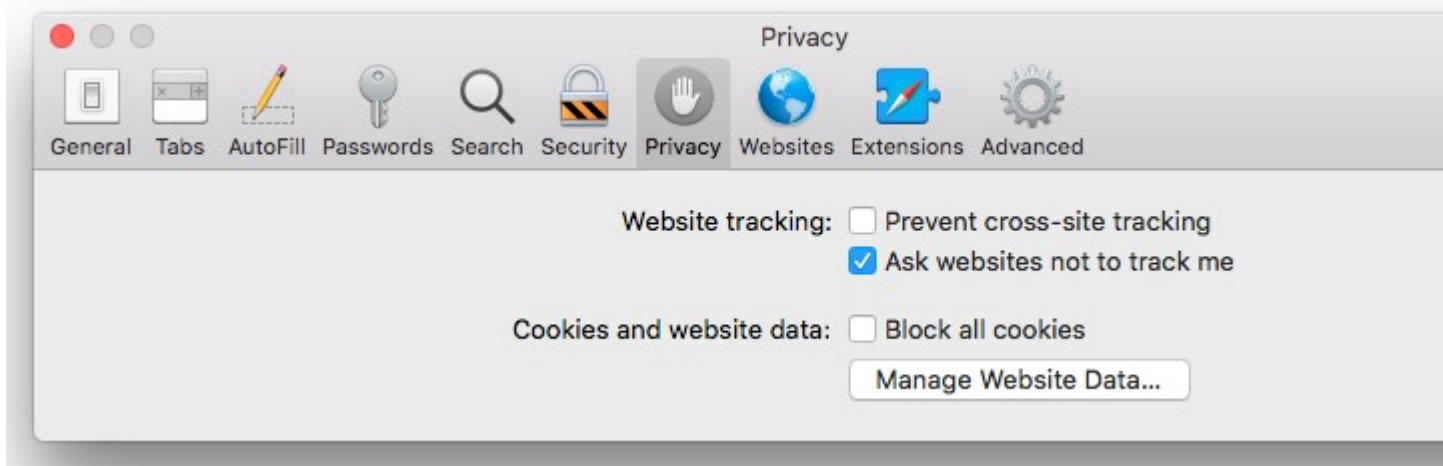
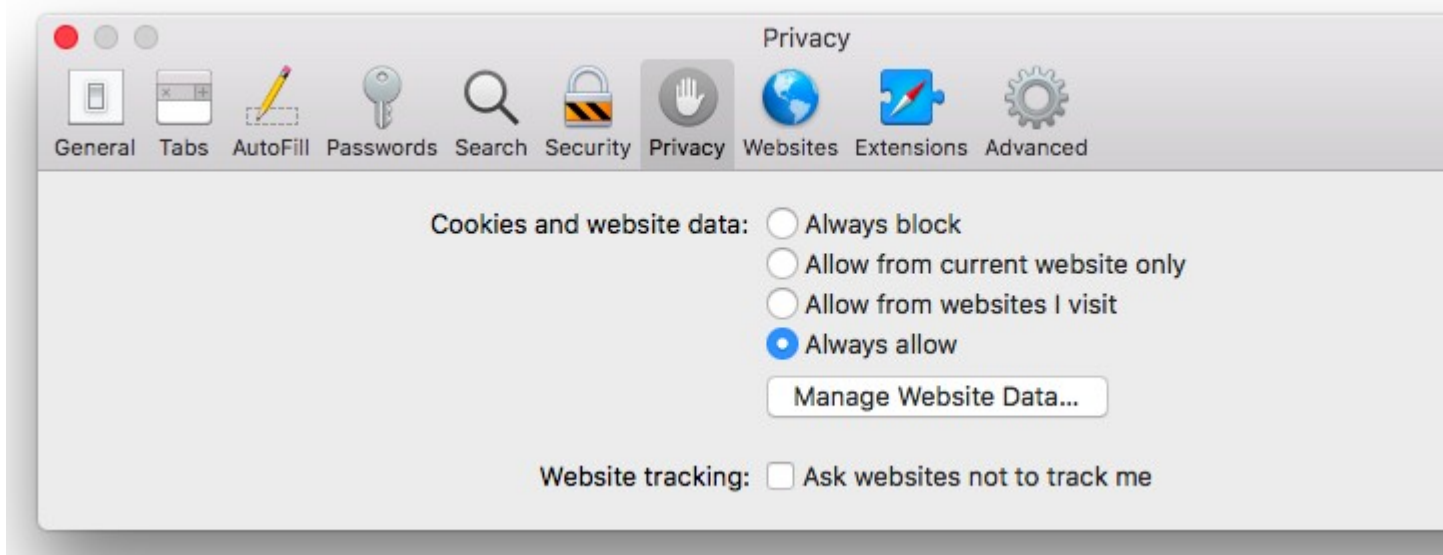
Integrated Terminal feature is using third-party cookie. Therefore, some Chrome extensions (such as Ad-blockers) or browser's configuration might block the request to the terminal console. For this reason, please disable such extensions or configuration.

### Fail to connect in Safari

If you are receiving the error message regarding cross-site tracking or cookie, please disable them in Safari's settings as follows:

1. From Safari, go to **Preferences** → **Privacy** .
2. The Privacy tab might look differently with different version. Please make sure that the cookies are unblocked and the cross-site tracking is allowed. Here is an example:





## Limitations

Certain commands and operations are restricted for security purpose. Please contact us if you find any issues.

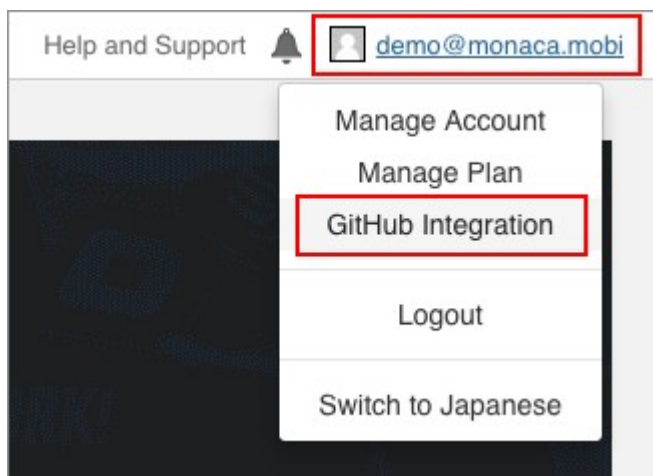
# GitHub Integration

Generally, you can only connect to public repositories. However, with a valid Monaca subscription plan, you can additionally connect to private repositories. Please refer to [Monaca Subscription Plan](#).

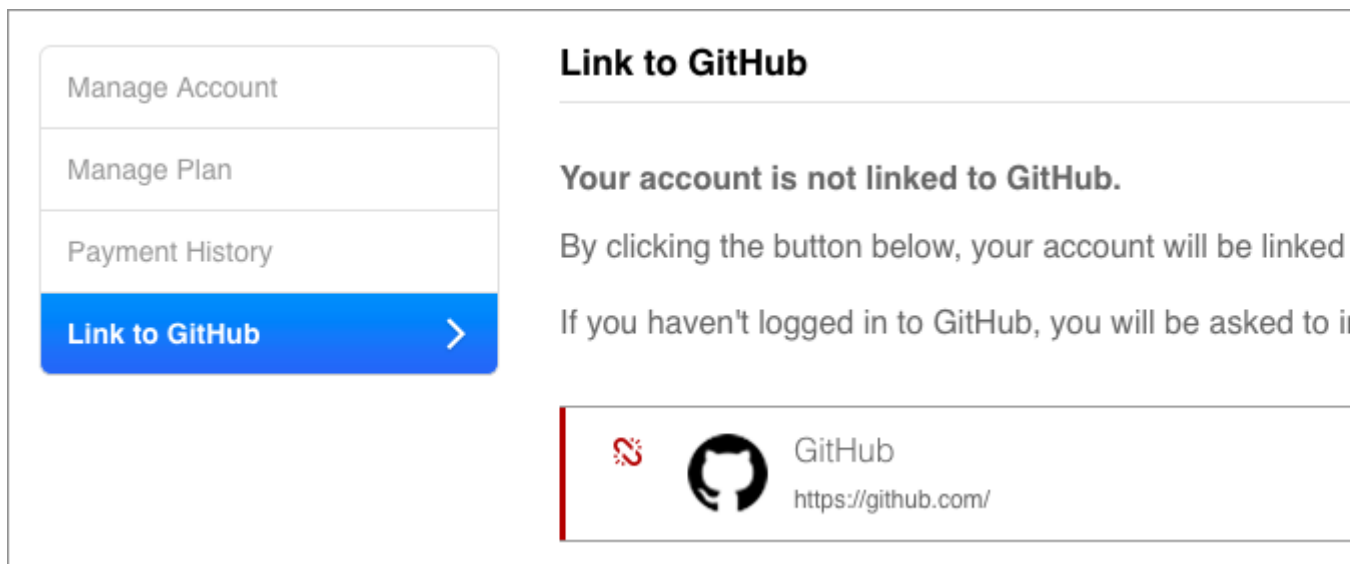
## Setup

In this section, you will learn how to link Monaca account to your GitHub account. Please proceed as follows:

1. Go to [Link to GitHub](#) page.



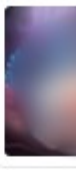
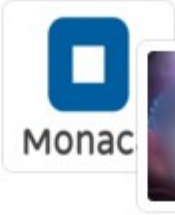
2. Click on Link button. You are required to login with your GitHub account before being redirected to GitHub's Authorize Application page.




3. In the Authorize Application page, you will be asked to authorize the application in order to link Monaca account to GitHub. Click on Authorize application to proceed.


# Authorize application


Monaca by @monaca would like permission to access your account



## Review permissions

**Personal user data**  
Full access

**Repository webhooks and services**  
Read and write access

**Repositories**  
Public and private

### Monaca


Monaca is a HTML5 development platform for Cordova and Ionic apps.

[Visit application's website](#)

[Learn more about O](#)

## Organization access

Organizations determine whether the application can access their data.

 **monaca** ✓

Authorize application

4. Now your Monaca account is successfully linked to your GitHub account.

Manage Account

Manage Plan



Payment History

Link to GitHub >


## Link to GitHub

**Your account is already linked to GitHub.**

By clicking the button below, your account will be unlinked.



GitHub  
<https://github.com/>

 Monaca  
demo@monaca.m

Please note that you can only link one GitHub account to a Monaca account. If you try to link with multiple accounts, you will encounter an error.

# Connecting a Monaca Project to New GitHub Repository

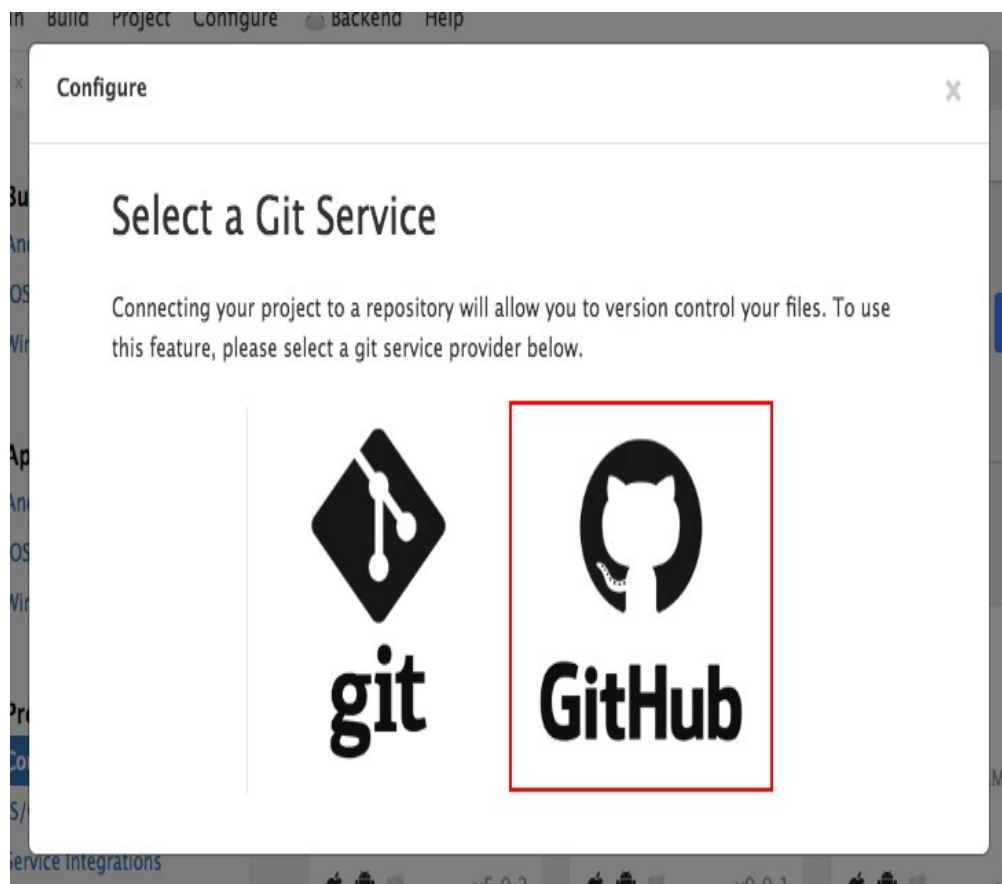
## Step 1: Creating a New Empty Repository

Go to your GitHub account and create a new empty repository (without Readme file).

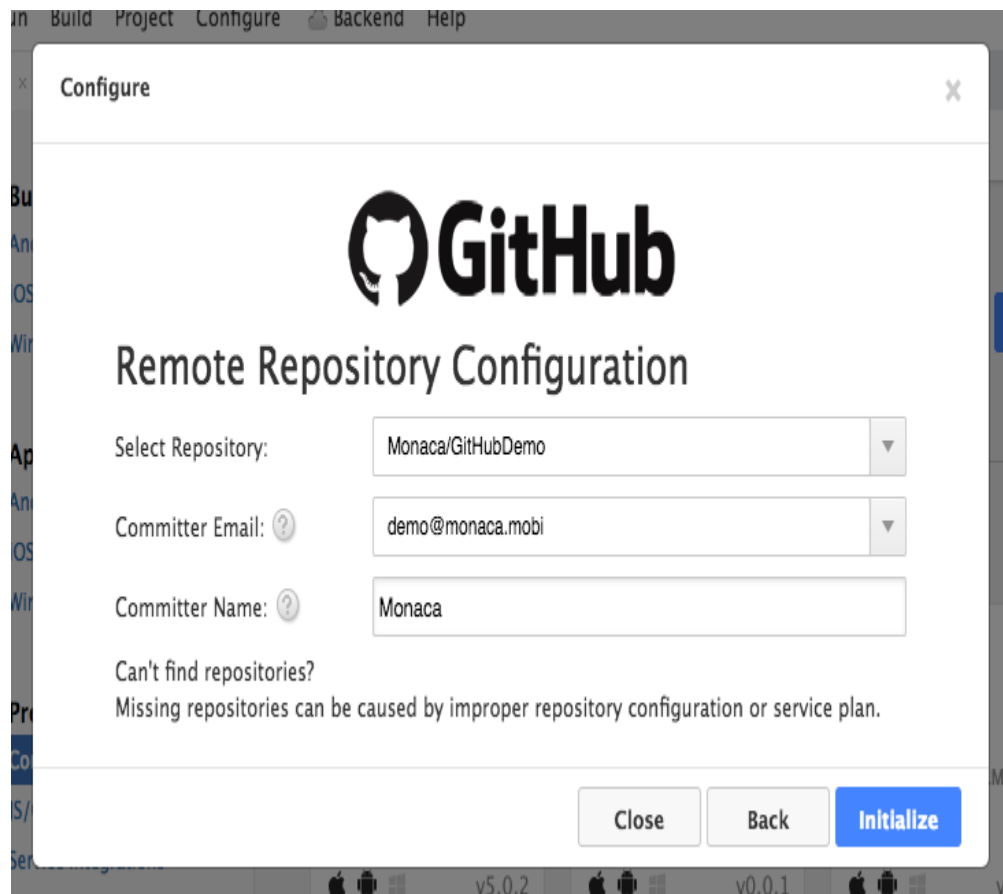
## Step 2: Connecting Monaca Project with the Repository

Assuming that you have successfully integrated your Monaca account to GitHub account, you need to do some configurations in Monaca Cloud IDE in order to connect your project to your repository:

1. From Monaca Dashboard, open a project you want to connect to a repository.
2. From Monaca Cloud IDE menu, go to **Project** → **VCS Configure**.
3. Select GitHub.



4. Select your remote empty repository. Then, click on Initialize button to save the configuration.



Repository cannot be changed after configured.

5. Your project is then being uploaded to your new repository in GitHub. By default, your working branch will be configured as `master`. If you want to switch to another working branch, please go to **Project** → **VCS Configure**.



The image shows a 'Configure' dialog box titled 'Remote Repository Configuration' with the GitHub logo. It contains the following fields and information:

- Current Repository:** [Monaca/GitHubDemo](#). Below it, a message states: 'Repositories can not be changed after setup.'
- Current Working Branch:** A dropdown menu showing 'dev'. This field is highlighted with a red rectangle.
- Committer Email:** A dropdown menu showing 'demo@monaca.mobi'.
- Committer Name:** A text field containing 'Monaca'.

Below the fields, a note reads: 'If your GitHub username and/or repository name changes, please click on Clear Cache & Save even if you made no changes.'

At the bottom right, there are two buttons: 'Close' and 'Clear Cache & Save'.

## Importing a Project from GitHub Repository into Monaca

Once you have linked your Monaca account with GitHub, you can import any existing projects from GitHub repositories to Monaca Cloud IDE.

1. From Dashboard, click on Import.
2. In Import Project dialog, fill in the necessary information and choose **Import from GitHub Repository**. Then, select the repository and click on Import button.

Configure backend help

### Import Project

**Project Name \***

GitHub Demo

**Description**

**Import Method**

☐ Import from URL

http://example.com/monaca-project.zip

☐ Upload Project Package

Choose File No file chosen

☐ Import from Git Repository

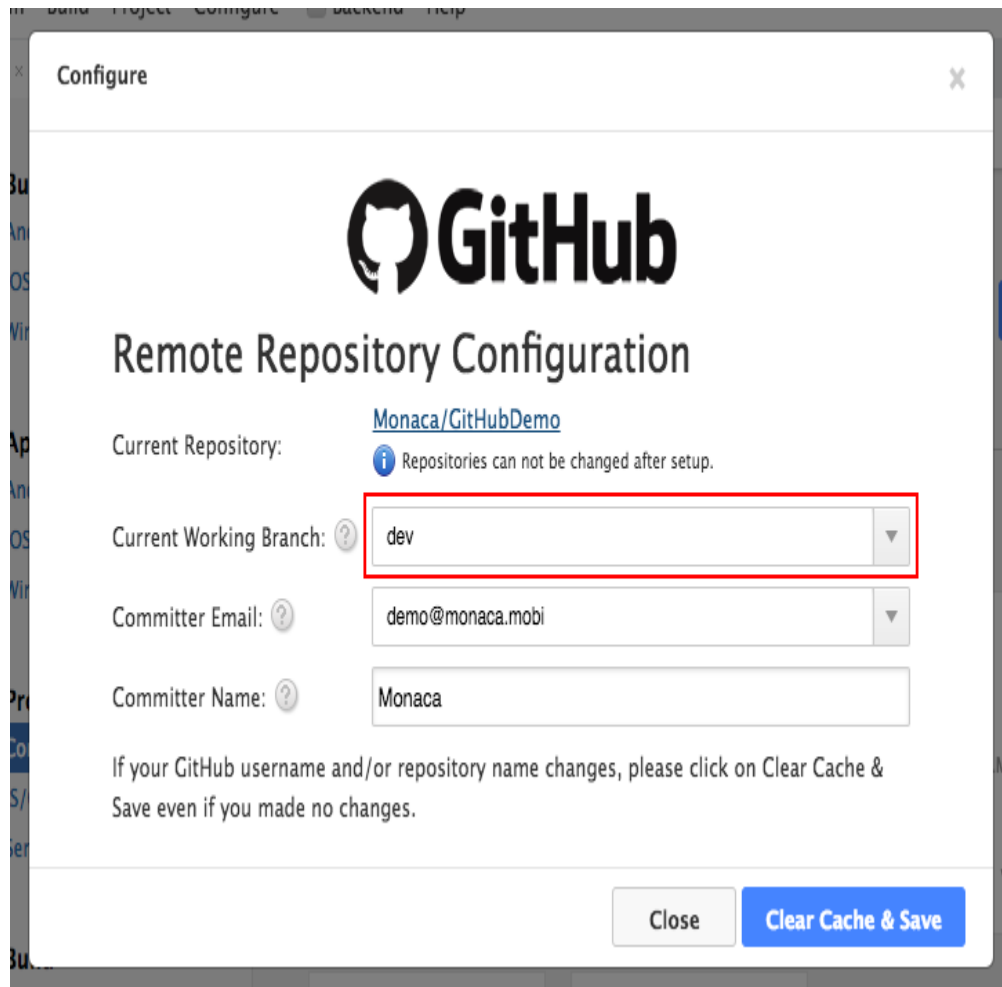
ssh://

☒ Import from GitHub Repository

asial/2018asialhp

Import Cancel

3. If the import is successful, the new project will be added to the Dashboard. By default, your working branch will be configured as `master`. If you want to switch to another working branch, please go to **Project** → **VCS Configure**.



## Working with Remotes

Once you have successfully connected your project with a repository, you can start working on the same project with your team members and keep it synchronized to the latest updates in Monaca Cloud IDE.

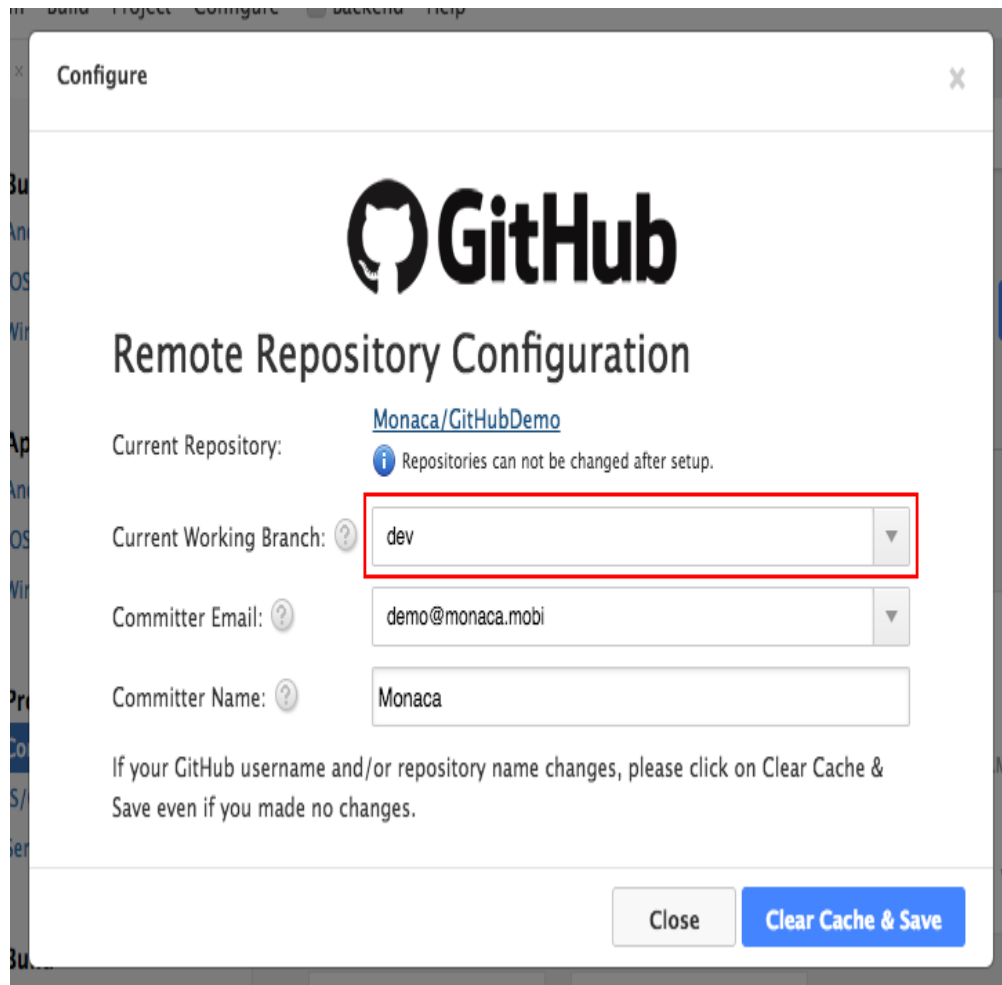
Monaca Cloud IDE provides a very user-friendly interface supporting version control of your code. Without learning Git from the scratch, you can still perform some basic Git commands directly from the IDE.

### Checkout Current Working Branch

If you have more than one branch and want to switch the current working branch, please do as follows:

1. From Monaca Cloud IDE menu, go to **Project** → **VCS Configure** .
2. Choose the branch you want to switch to from current working branch, and click Clear Cache & Save button.





You will get an error message if you try to checkout to a new branch while you have not committed changes on the current working branch yet. Please push your changes to current branch first before checking out to another branch.

## Retrieve Changes from Remotes

If you are familiar with Git commands, retrieve changes from remotes here refers to `git pull` command. In order to retrieve the changes from remotes, from Monaca Cloud IDE menu, go to **Project** → **Pull**. If there are changes in your remotes, you will receive those updates in your working branch.

When you create a project in IDE, Monaca keeps all the files in `www` folder. Files outside of this folder are also synced even though they will not appear in the IDE.

## Commit Changes to Remotes

After making changes in the current working branch, you are able to commit them back to your remote repository. To commit your changes, please do as follows:

1. From Monaca Cloud IDE menu, go to **Project** → **Commit**.
2. Fill in your commit message and check files you want to commit. Then, click on Commit button.

3. Once you are ready to push those updates to the remote repository, select **Project → Push** . Once it is successfully pushed, your remotes will contain latest changes made in Monaca Cloud IDE.

## Show Remote Commit History

In order to see full history of your previous commits in remote repository, go to **Project → Show Remote History** . You will be redirected to the remote Git service provider's website showing a commit history of your current branch.

## Show Local Commit History

Every commit you made in your Monaca Cloud IDE is called Local Commit. You are able to view your local commit history through **Project → Show Commit History** . To view the changes in each commit, click on the commit on the left panel.

## Unlink Monaca Account from GitHub

In order to to unlink your Monaca account from GitHub, please do as follows:

1. Go to [Link to GitHub](#) page.
2. Click on Unlink button.

Manage Account

Manage Plan


Payment History


Link to GitHub >

### Link to GitHub


**Your account is already linked to GitHub.**

By clicking the button below, your account will be unlinked from





GitHub  
<https://github.com/>



Monaca  
demo@monaca.mobi

## Unable to Re-link to GitHub

If you accidentally revoke Monaca access with Github account, attempt to re-link with the steps described in [Setup](#) will not be possible. You will see the following error:

Manage Account

Manage Plan

Payment History

Link to GitHub >

### Link to GitHub


**Your account is already linked to GitHub.**

By clicking the button below, your account will be unlinked from GitHub.


Unable to fetch information from GitHub. Please sign out of Monaca and link your GitHub account.

Therefore, in order to re-link your account in this case, please do as follows:


1. Logout from Monaca Cloud IDE.
2. Go to [Monaca Login](#) page and choose Sign in with GitHub.




Login

 Sign In With GitHub

or

 E-mail address

 Password

☐ Keep my account logged in

Login

[Forgot password?](#)




3. Fill in your GitHub account information.
4. Then, you will be redirected to GitHub's Authorize Application page. Click on Authorize application to proceed.

# Authorize application

Monaca by @monaca would like permission to access your account



## Review permissions

	<b>Personal user data</b> Full access	▼
	<b>Repository webhooks and services</b> Read and write access	▼
	<b>Repositories</b> Public and private	▼

## Monaca

Monaca is a HTML5 development platform for Cordova and Ionic apps.

[Visit application's website](#)

[Learn more about O](#)

## Organization access

Organizations determine whether the application can access their data.

	<b>monaca</b> ✓
---	-----------------

**Authorize application**

5. After this, your Monaca account should be successfully linked to GitHub account. You can confirm in the Link to GitHub page. It should appear like this:

Manage Account

Manage Plan

Payment History

**Link to GitHub**



## Link to GitHub

**Your account is already linked to GitHub.**

By clicking the button below, your account will be unlinked.



GitHub  
<https://github.com/>



Monaca  
demo@monaca.m

# Git SSH Integration

Monaca allows you to connect your project with a repository using any Git services through SSH authentication. GitHub is one of Git services supporting SSH connection. Therefore, we will be using GitHub as an example in the following subsections.

Generally, you can only connect to public repositories. However, with valid Monaca subscription plan, you can additionally connect to private repositories. Please refer to [Monaca Subscription Plan](#).

## Prerequisites

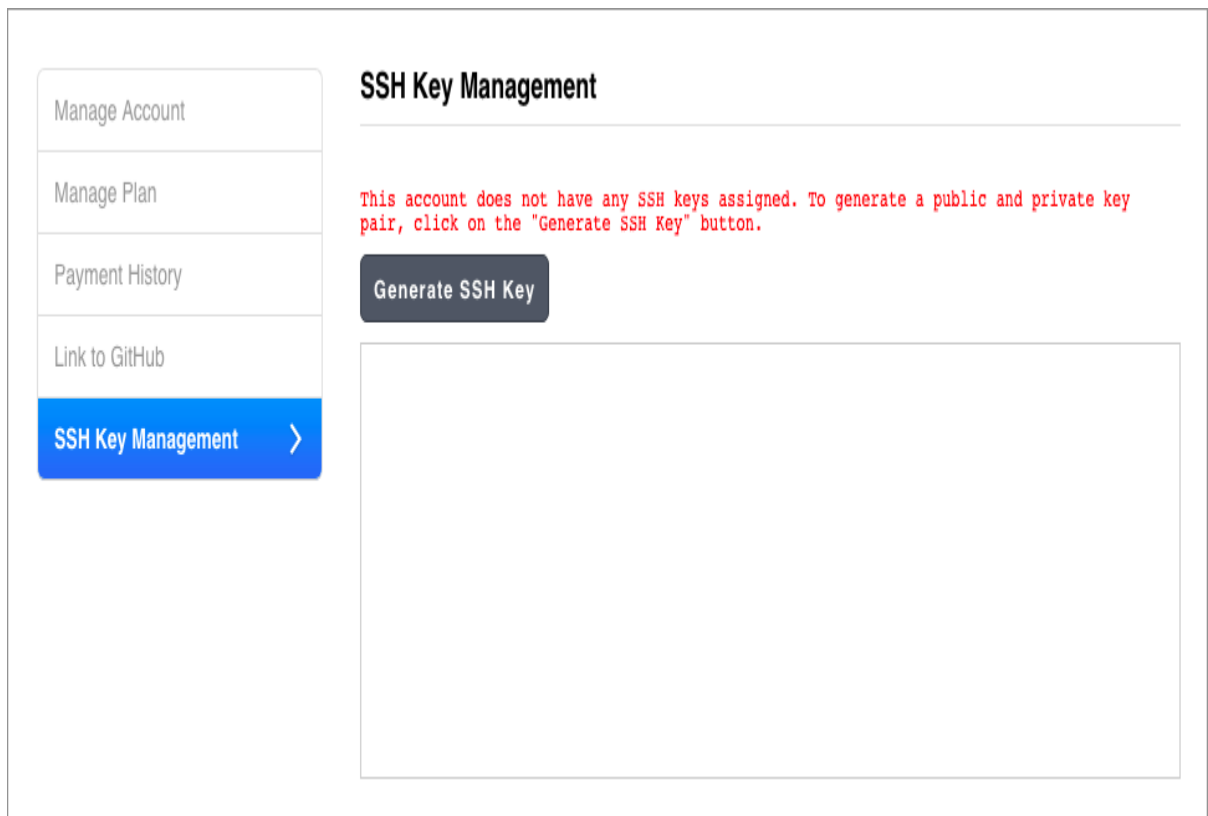
In order to use version control in Monaca Cloud IDE, you will need:

- an account from any Git services with SSH support

## Setup

### Step 1: Generating SSH Key

1. Go to [SSH Key Management](#) page.
2. Click on Generate SSH Key button.



3. Copy the generated key which will be used in the next section ([Adding the SSH Key to Git Service](#)).

## Step 2: Adding the SSH Key to Git Service

For proper authentication between Monaca and your Git service provider, the generated SSH key from the previous section needs to be added to your Git service account.

In this example, we are using GitHub. For other services, please review your Git services' documentation for support.

1. Go to [SSH Key](#) page.
2. Click on New SSH key button.
3. Fill in the following information:
  - **Title:** Title of the key
  - **Key:** The SSH key generated from the Monaca account earlier (as shown in the above section)
4. Then, click on Add SSH key button.

## Connecting a Monaca Project to New Git Repository

### Step 1: Creating a New Empty Repository

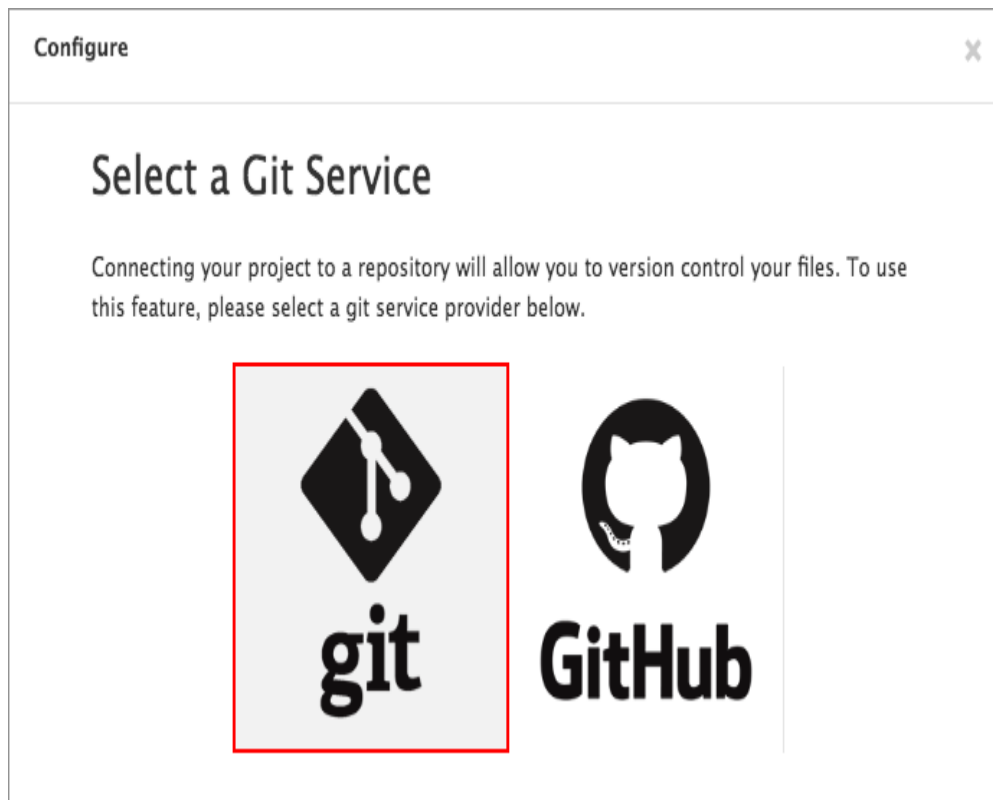
You can create a new repository in any Git services. In this case, we are creating a new repository in GitHub as follows:

1. Go to your GitHub account and create a new empty repository (without Readme file).
2. Then, select the SSH button.
3. Copy the SSH link (see below screenshot as example). We will use this link in the Monaca Cloud IDE later in the next section.



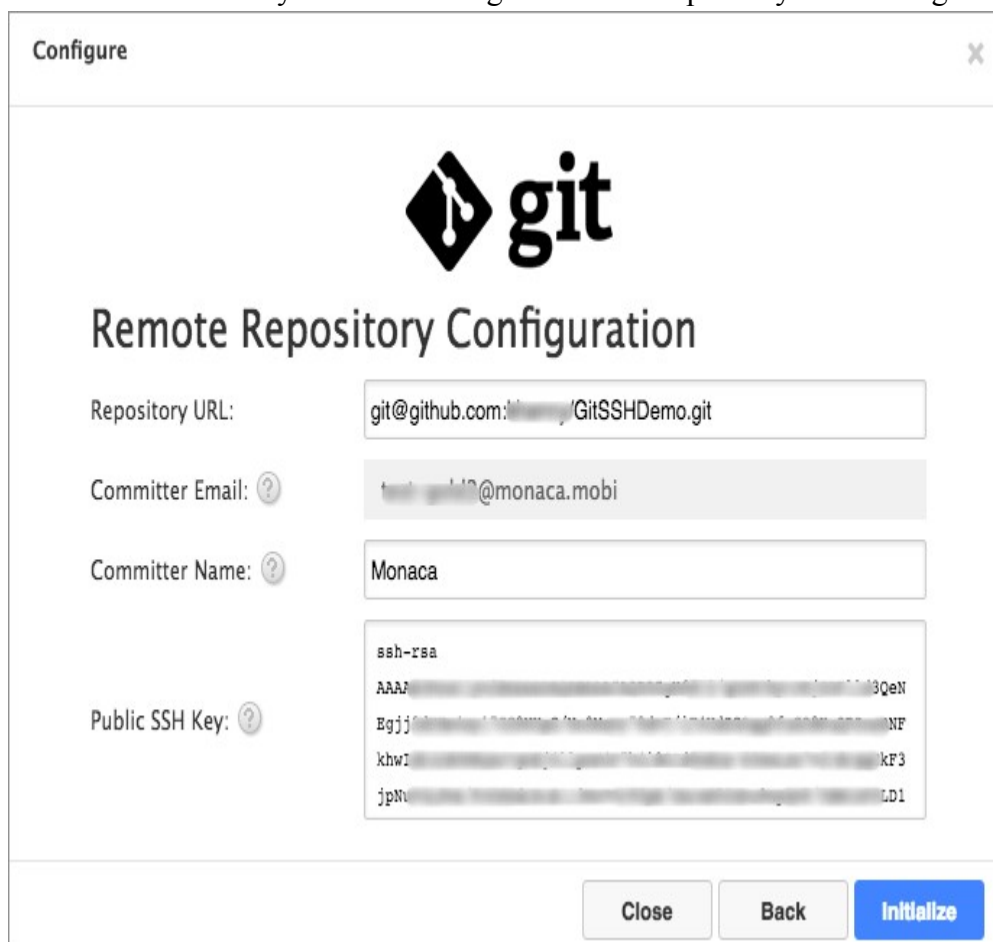
## Step 2: Connecting Monaca Project with Repository

1. From Monaca Dashboard, open a project you want to connect to a repository.
2. From Monaca Cloud IDE menu, go to **Project** → **VCS Configure**.
3. Select the **Git SSH** option as shown below:



4. Then, you will need to input the **Repository URL** (the link you found in the above section) and **Committer Name**.

Please be aware that you cannot change to another repository after configuration.

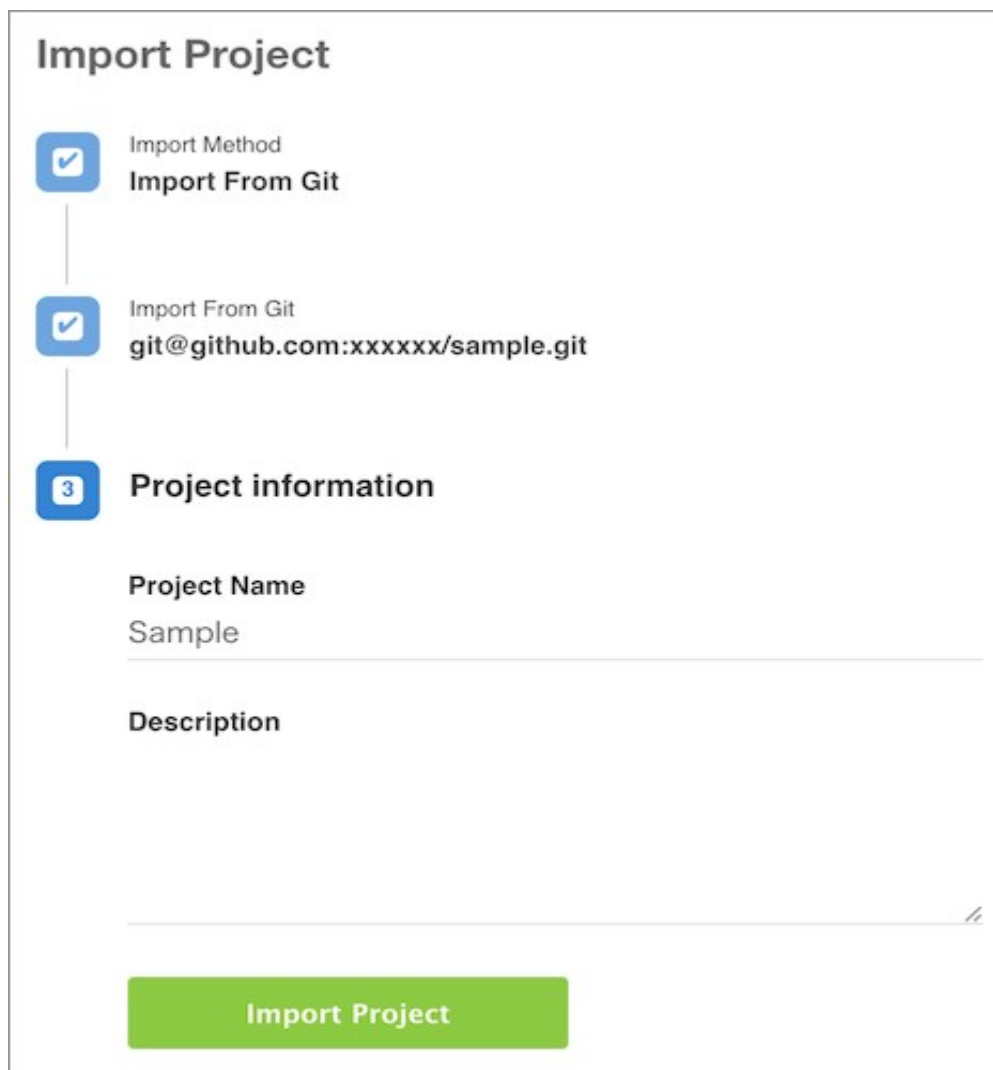




5. Click on Initialize button.
6. Your project is then being uploaded to your new repository. By default, your working branch will be configured as **master**. If you want to switch to another working branch, please go to **Project** → **VCS Configure**.

## Importing a Project from Git Repository into Monaca

1. Log into [Monaca Cloud IDE](#) with your Monaca account.
2. From Dashboard, go to Import.
3. Select **Import from Git**. Then, input the necessary information such as the repository URL and click the Import Project button.



The screenshot shows the 'Import Project' form in Monaca Cloud IDE. It has a vertical progress indicator on the left with three steps: 1. 'Import Method' (checked), 2. 'Import From Git' (checked), and 3. 'Project information' (active). The 'Project information' section contains a 'Project Name' field with the value 'Sample' and a 'Description' field which is empty. At the bottom is a green 'Import Project' button.

**Import Project**

☒ Import Method  
**Import From Git**

☒ Import From Git  
git@github.com:xxxxxx/sample.git

**3** **Project information**

**Project Name**  
Sample

**Description**

**Import Project**

4. If the import is successful, the new project will be added to Monaca Dashboard. By default, your working branch will be configured as **master**. If you want to switch to another working branch, please go to **Project** → **VCS Configure**.

## Working with Remotes

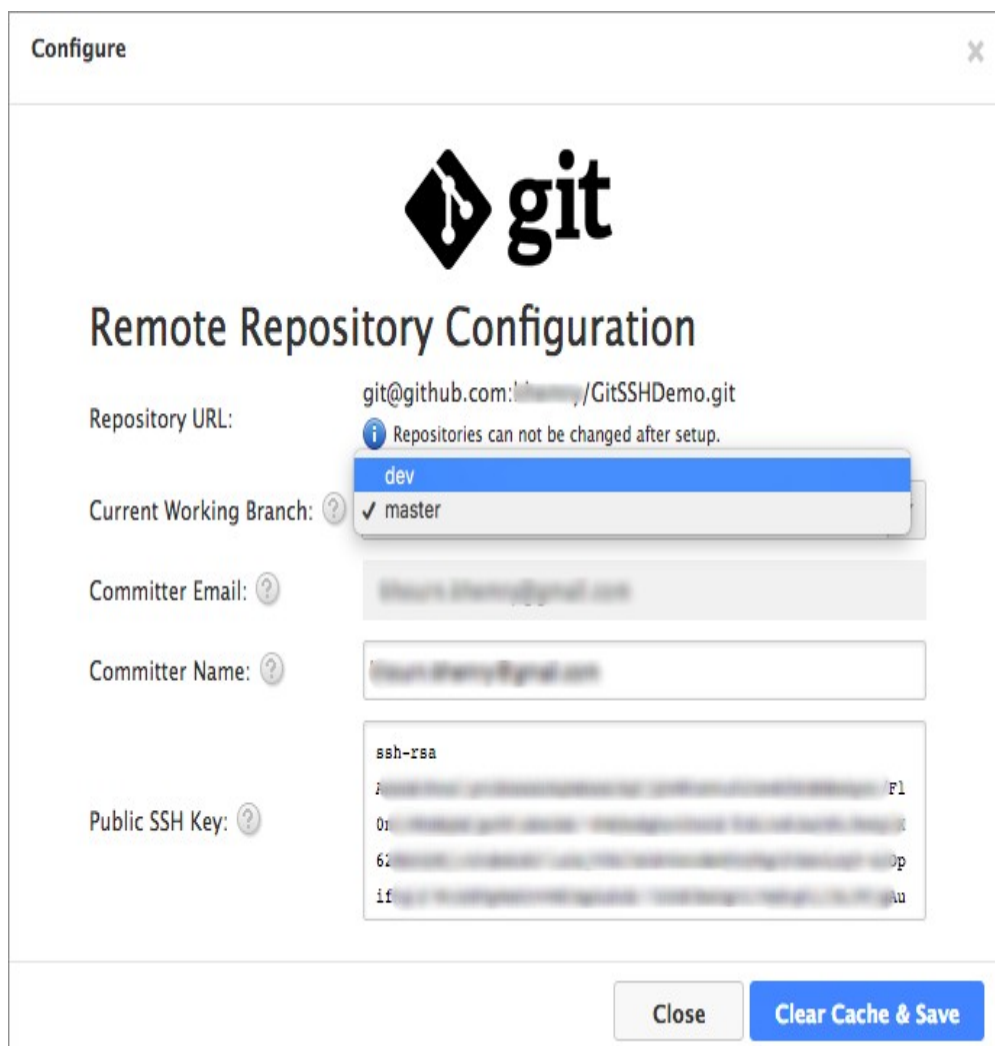
Once you have successfully connected your project with a repository, you can start working on the same project with your team members and keep it synchronized to the latest updates in Monaca Cloud IDE.

Monaca Cloud IDE provides a very user-friendly interface supporting version control of your code. Without learning Git from the scratch, you can still perform some basic Git commands directly from the IDE.

### Checkout Current Working Branch

If you have more than one branch and want to switch the current working branch, please do as follows:

1. From Monaca Cloud IDE menu, go to **Project** → **VCS Configure**.
2. Choose the branch you want to switch to from current working branch, and click Clear Cache & Save button.



You will get an error message if you try to checkout to a new branch while you have not committed changes on the current working branch yet. Please push your changes to current branch first before checking out to another branch.

## Retrieve Changes from Remotes

If you are familiar with Git commands, retrieve changes from remotes here refers to `git pull` command. In order to retrieve the changes from remotes, from Monaca Cloud IDE menu, go to **Project → Pull** . If there are changes in your remotes, you will receive those updates in your working branch.

When you create a project in IDE, Monaca keeps all the files in `www` folder. Files outside of this folder are also synced even though they will not appear in the IDE.

## Commit Changes to Remotes

After making changes in the current working branch, you are able to commit them back to your remote repository. To commit your changes, please do as follows:

1. From Monaca Cloud IDE menu, go to Version **Project → Commit** .
2. Fill in your commit message and check files you want to commit. Then, click on Commit button.
3. Once you are ready to push those updates to the remote repository, select **Project → Push** . Once it is successfully pushed, your remotes will contain latest changes made in Monaca Cloud IDE.

## Show Remote Commit History

Since there are numerous public and private Git services available to choose from and the URL to the Remote Commit History is not known for all services, this option is not available for projects linked to a repository with SSH.

In order to see full history of your previous commits in remote repository, go to **Project → Show Remote History** . You will be redirected to the remote Git service provider's website showing a commit history of your current branch.

## Show Local Commit History

Every commit you made in your Monaca Cloud IDE is called Local Commit. You are able to view your local commit history through **Project → Show Commit History** . To view the changes in each commit, click on the commit on the left panel.



