

Get started

Samples & tutorials

Development

- ▶ [User interface](#)
- ▶ [Data & backend](#)
- ▶ [Accessibility & internationalization](#)
- ▶ [Platform integration](#)
- ▶ [Packages & plugins](#)
- ▶ [Add Flutter to existing app](#)
- ▶ [Tools & techniques](#)
- ▶ [Migration notes](#)

Testing & debugging

Performance & optimization

Deployment

- [Obfuscating Dart code](#)
- [Creating flavors for Flutter](#)
- [Build and release an Android app](#)
- [Build and release an iOS app](#)
- [Build and release a web app](#)
- [Continuous deployment](#)

Resources

Reference

- [Widget index](#)
- [API reference](#)
- [Package site](#)

Build and release an iOS app

Contents

- [Preliminaries](#)
- [Register your app on App Store Connect](#)
 - [Register a Bundle ID](#)
 - [Create an application record on App Store Connect](#)
- [Review Xcode project settings](#)
- [Updating the app’s name](#)
- [Updating the app’s version number](#)
- [Add an app icon](#)
- [Create a build archive](#)
- [Release your app on TestFlight](#)
- [Release your app to the App Store](#)
- [Troubleshooting](#)

This guide provides a step-by-step walkthrough of releasing a Flutter app to the [App Store](#) and [TestFlight](#).

🔗 Preliminaries

Before beginning the process of releasing your app, ensure that it meets Apple’s [App Review Guidelines](#).

In order to publish your app to the App Store, you must first enroll in the [Apple Developer Program](#). You can read more about the various membership options in Apple’s [Choosing a Membership](#) guide.

Register your app on App Store Connect

Manage your app’s life cycle on [App Store Connect](#) (formerly iTunes Connect). You define your app name and description, add screenshots, set pricing, and manage releases to the App Store and TestFlight.

Registering your app involves two steps: registering a unique Bundle ID, and creating an application record on App Store Connect.

For a detailed overview of App Store Connect, see the [App Store Connect](#) guide.

Register a Bundle ID

Every iOS application is associated with a Bundle ID, a unique identifier registered with Apple. To register a Bundle ID for your app follow these steps:

1. Open the [App IDs](#) page of your developer account.
2. Click **+** to create a new Bundle ID.
3. Enter an app name, select **Explicit App ID**, and enter an ID.
4. Select the services your app uses, then click **Continue**.
5. On the next page, confirm the details and click **Register** to register your Bundle ID.

Create an application record on App Store Connect

Register your app on App Store Connect:

1. Open [App Store Connect](#) in your browser.
2. On the App Store Connect landing page, click **My Apps**.
3. Click **+** in the top-left corner of the My Apps page, then select **New App**.
4. Fill in your app details in the form that appears. In the Platforms section, ensure that iOS is checked. Since Flutter does not currently support tvOS, leave that checkbox unchecked. Click **Create**.
5. Navigate to the application details for your app and select **App Information** from the sidebar.
6. In the General Information section, select the Bundle ID you registered in the preceding step.

For a detailed overview, see [Add an app to your account](#).

Review Xcode project settings

This step covers reviewing the most important settings in the Xcode workspace. For detailed procedures and descriptions, see [Prepare for app distribution](#).

[Get started](#)

[Samples & tutorials](#)

[Development](#)

- ▶ [User interface](#)
- ▶ [Data & backend](#)
- ▶ [Accessibility & internationalization](#)
- ▶ [Platform integration](#)
- ▶ [Packages & plugins](#)
- ▶ [Add Flutter to existing app](#)
- ▶ [Tools & techniques](#)
- ▶ [Migration notes](#)

[Testing & debugging](#)

[Performance & optimization](#)

[Deployment](#)

- [Obfuscating Dart code](#)
- [Creating flavors for Flutter](#)
- [Build and release an Android app](#)
- [Build and release an iOS app](#)
- [Build and release a web app](#)
- [Continuous deployment](#)

[Resources](#)

[Reference](#)

- [Widget index](#)
- [API reference](#)
- [Package site](#)

Navigate to your target’s settings in Xcode:

1. In Xcode, open `Runner.xcworkspace` in your app’s `ios` folder.
2. To view your app’s settings, select the **Runner** project in the Xcode project navigator. Then, in the main view sidebar, select the **Runner** target.
3. Select the **General** tab.

Verify the most important settings.

In the **Identity** section:

Display Name

Runner

Changing the display name to anything other than `Runner` causes the app not to run with the Flutter tools. For more information, see [Issue 41793](#).

Bundle Identifier

The App ID you registered on App Store Connect.

In the **Signing** section:

Automatically manage signing

Whether Xcode should automatically manage app signing and provisioning. This is set `true` by default, which should be sufficient for most apps. For more complex scenarios, see the [Code Signing Guide](#).

Team

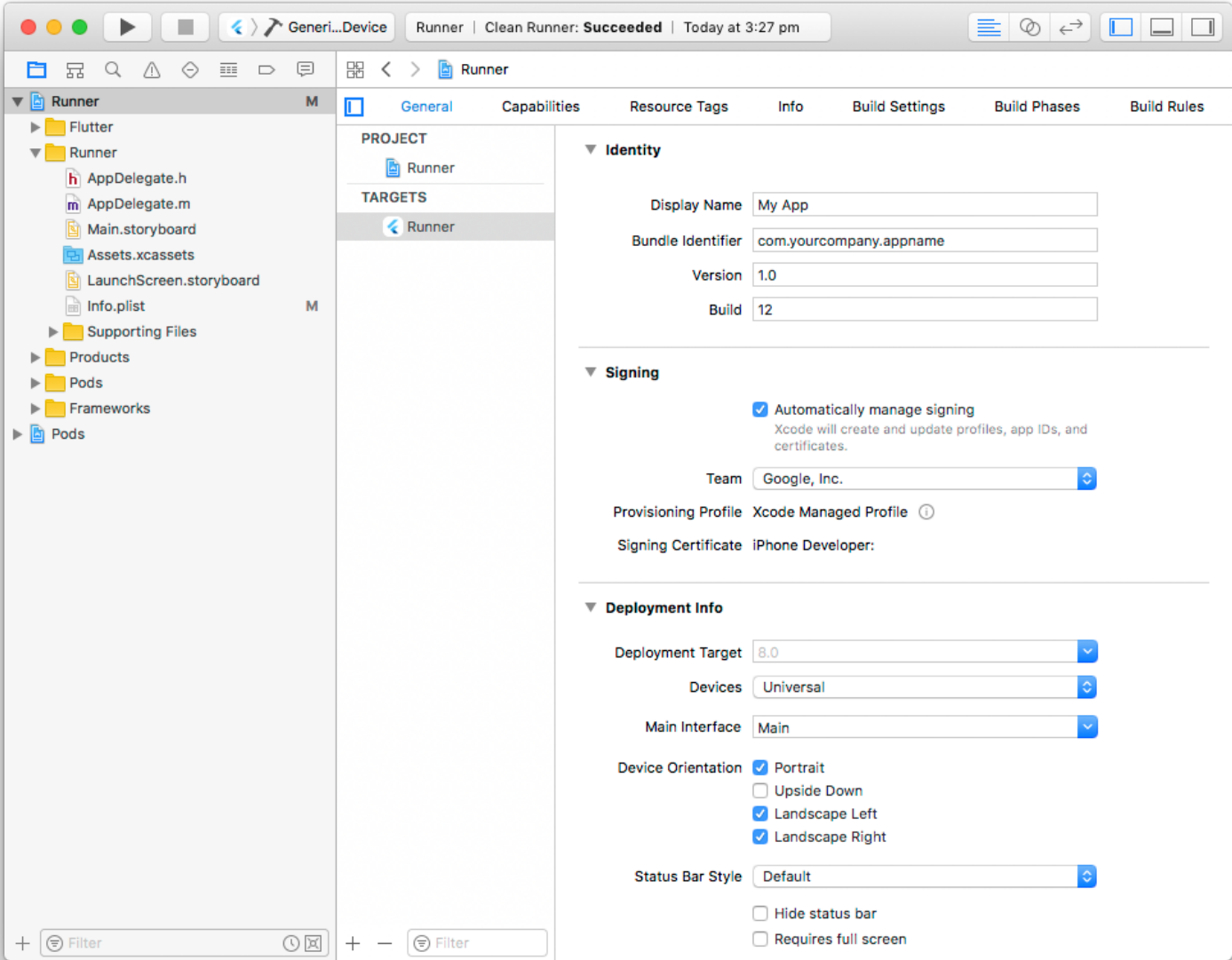
Select the team associated with your registered Apple Developer account. If required, select **Add Account...**, then update this setting.

In the **Deployment Info** section:

Deployment Target:

The minimum iOS version that your app supports. Flutter supports iOS 8.0 and later. If your app includes Objective-C or Swift code that makes use of APIs that were unavailable in iOS 8, update this setting appropriately.

The **General** tab of your project settings should resemble the following:



For a detailed overview of app signing, see [Create, export, and delete signing certificates](#).

Updating the app’s name

In the main view sidebar, select the `Info.plist` file. Rename the **Bundle name** field.

Updating the app’s version number

[Get started](#)

[Samples & tutorials](#)

[Development](#)

- ▶ [User interface](#)
- ▶ [Data & backend](#)
- ▶ [Accessibility & internationalization](#)
- ▶ [Platform integration](#)
- ▶ [Packages & plugins](#)
- ▶ [Add Flutter to existing app](#)
- ▶ [Tools & techniques](#)
- ▶ [Migration notes](#)

[Testing & debugging](#)

[Performance & optimization](#)

[Deployment](#)

- [Obfuscating Dart code](#)
- [Creating flavors for Flutter](#)
- [Build and release an Android app](#)
- [Build and release an iOS app](#)
- [Build and release a web app](#)
- [Continuous deployment](#)

[Resources](#)

[Reference](#)

- [Widget index](#)
- [API reference](#)
- [Package site](#)

The default version number of the app is `1.0.0`. To update it, navigate to the `pubspec.yaml` file and update the following line:

```
version: 1.0.0+1
```

The version number is three numbers separated by dots, such as `1.0.0` in the example above, followed by an optional build number such as `1` in the example above, separated by a `+`.

Both the version and the build number may be overridden in Flutter’s build by specifying `--build-name` and `--build-number`, respectively.

In iOS, `build-name` uses `CFBundleShortVersionString` while `build-number` uses `CFBundleVersion`. Read more about iOS versioning at [Core Foundation Keys](#) on the Apple Developer’s site.

Add an app icon

When a new Flutter app is created, a placeholder icon set is created. This step covers replacing these placeholder icons with your app’s icons:

1. Review the [iOS App Icon](#) guidelines.
2. In the Xcode project navigator, select `Assets.xcassets` in the `Runner` folder. Update the placeholder icons with your own app icons.
3. Verify the icon has been replaced by running your app using `flutter run`.

Create a build archive

This step covers creating a build archive and uploading your build to App Store Connect.

During development, you’ve been building, debugging, and testing with *debug* builds. When you’re ready to ship your app to users on the App Store or TestFlight, you need to prepare a *release* build. At this point, you might consider [obfuscating your Dart code](#) to make it more difficult to reverse engineer. Obfuscating your code involves adding a couple flags to your build command.

On the command line, follow these steps in your application directory:

1. Run `flutter build ios` to create a release build (`flutter build` defaults to `--release`).
2. To ensure that Xcode refreshes the release mode configuration, close and re-open your Xcode workspace. For Xcode 8.3 and later, this step is not required.

In Xcode, configure the app version and build:

1. In Xcode, open `Runner.xcworkspace` in your app’s `ios` folder.
2. Select **Product > Scheme > Runner**.
3. Select **Product > Destination > Generic iOS Device**.
4. Select **Runner** in the Xcode project navigator, then select the **Runner** target in the settings view sidebar.
5. In the Identity section, update the **Version** to the user-facing version number you wish to publish.
6. In the Identity section, update the **Build** identifier to a unique build number used to track this build on App Store Connect. Each upload requires a unique build number.

Finally, create a build archive and upload it to App Store Connect:

1. Select **Product > Archive** to produce a build archive.
2. In the sidebar of the Xcode Organizer window, select your iOS app, then select the build archive you just produced.
3. Click the **Validate App** button. If any issues are reported, address them and produce another build. You can reuse the same build ID until you upload an archive.
4. After the archive has been successfully validated, click **Distribute App**. You can follow the status of your build in the Activity tab of your app’s details page on [App Store Connect](#).

You should receive an email within 30 minutes notifying you that your build has been validated and is available to release to testers on TestFlight. At this point you can choose whether to release on TestFlight, or go ahead and release your app to the App Store.

For more details, see [Upload an app to App Store Connect](#).

Release your app on TestFlight

[TestFlight](#) allows developers to push their apps to internal and external testers. This optional step covers releasing your build on TestFlight.

1. Navigate to the TestFlight tab of your app’s application details page on [App Store Connect](#).
2. Select **Internal Testing** in the sidebar.
3. Select the build to publish to testers, then click **Save**.
4. Add the email addresses of any internal testers. You can add additional internal users in the **Users and Roles** page of App Store Connect, available from the dropdown menu at the top of the page.

For more details, see [Distribute an app using TestFlight](#).

[Get started](#)

[Samples & tutorials](#)

[Development](#)

- ▶ [User interface](#)
- ▶ [Data & backend](#)
- ▶ [Accessibility & internationalization](#)
- ▶ [Platform integration](#)
- ▶ [Packages & plugins](#)
- ▶ [Add Flutter to existing app](#)
- ▶ [Tools & techniques](#)
- ▶ [Migration notes](#)



Release your app to the App Store

When you’re ready to release your app to the world, follow these steps to submit your app for review and release to the App Store

1. Select **Pricing and Availability** from the sidebar of your app’s application details page on [App Store Connect](#) and complete the required information.
2. Select the status from the sidebar. If this is the first release of this app, its status is **1.0 Prepare for Submission**. Complete the required fields.
3. Click **Submit for Review**.

Apple notifies you when their app review process is complete. Your app is released according to the instructions you specified in the **Version Release** section.

For more details, see [Distribute an app through the App Store](#).

Troubleshooting

The [Distribute your app](#) guide provides a detailed overview of the process of releasing an app to the App Store.



[flutter-dev@](#) • [terms](#) • [security](#) • [privacy](#) • [español](#) • [社区中文资源](#)

Except as otherwise noted, this work is licensed under a Creative Commons Attribution 4.0 International License, and code samples are licensed under the BSD License.

[Obfuscating Dart code](#)

[Creating flavors for Flutter](#)

[Build and release an Android app](#)

[Build and release an iOS app](#)

[Build and release a web app](#)

[Continuous deployment](#)

[Resources](#)



[Reference](#)

[Widget index](#)

[API reference](#) 

[Package site](#) 