EventRight Pro Add-on Flutter Apps

Welcome to EventRight Pro

Firstly, a huge thanks for purchasing our product, your support is truly appreciated!

We are more than happy to assist with any queries you have. this documentation is for a basic overview and about how to generate builds using this. Read this document thoroughly if you are experiencing any difficulties.

Kind Note: This is a Codebase to build your app upon (Not a No-code Drag n' Drop App builder)

Mobile Configuration & Build Generation

Prerequisites

- Flutter **v3.10.2 (EXACT VERSION)**
  https://docs.flutter.dev/release/archive?tab=windows
- Git for Windows
  https://git-scm.com/downloads
- Android Studio Chipmunk
  https://developer.android.com/studio/archive
- Emulator: **Pixel 5**
- XCode (for MacOS)
- Knowledge of above mentioned topics

Application Introduction

This is an add-on for the EventRight Pro Product, built with Flutter Cross Platform Framework. So they will work on Android & iOS as well.

For the Best User & Development Experience we recommand you use the complete package from provided by us instead of one app from our store and other apps from someone else. Because our backend & apps shares the API already. & It will be convient on update release as well.

Setting up Flutter

For Windows Users :-

System requirements

To install and run Flutter, your development environment must meet these minimum requirements:

- **Operating Systems**: Windows 10 or later (64-bit), x86-64 based.
- **Disk Space**: 1.64 GB (does not include disk space for IDE/tools).
- **Tools**: Flutter depends on these tools being available in your environment.
    - Windows PowerShell 5.0 or newer (this is pre-installed with Windows 10)
    - Git for Windows 2.x, with the **Use Git from the Windows Command Prompt** option.
      If Git for Windows is already installed, make sure you can run `git` commands from the command prompt or PowerShell.

Get the Flutter SDK

1- Download the following installation bundle to get the **3.16.4 Stable** release of the Flutter SDK:

> ⓘ Download Flutter SDK v3.16.4

2- Extract the zip file and place the contained `flutter` in the desired installation location for the Flutter SDK (for example, **C:\src\flutter**).

You are now ready to run Flutter commands in the Flutter Console.

Update your path

If you wish to run Flutter commands in the regular Windows console, take these steps to add Flutter to the `PATH` environment variable:

- From the Start search bar, enter ‘env’ and select **Edit environment variables for your account**.
- Under **User variables** check if there is an entry called **Path**:
    - If the entry exists, append the full path to `flutter\bin` using `;` as a separator from existing values.
    - If the entry doesn’t exist, create a new user variable named `Path` with the full path to `flutter\bin` as its value.

You have to close and reopen any existing console windows for these changes to take effect.

Run `flutter doctor`

From a console window that has the Flutter directory in the path (see above), run the following command to see if there are any platform dependencies you need to complete the setup:

```
C:\src\flutter>flutter doctor
```

This command checks your environment and displays a report of the status of your Flutter installation. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

For example:

```
[-] Android toolchain - develop for Android devices
    • Android SDK at D:\Android\sdk
      Android SDK is missing command line tools; download from https://goo.gl/XxQghQ
    • Try re-installing or updating your Android SDK,
      visit https://flutter.dev/setup/#android-setup for detailed instructions.
```

The following sections describe how to perform these tasks and finish the setup process. Once you have installed any missing dependencies, you can run the `flutter doctor` command again to verify that you've set everything up correctly.

For iOS Users :-

To install and run Flutter, your development environment must meet these minimum requirements:

System requirements

- **Operating Systems**: macOS (64-bit)
- **Disk Space**: 2.8 GB (does not include disk space for IDE/tools).
- **Tools**: Flutter uses `git` for installation and upgrade. We recommend installing Xcode, which includes `git`, but you can also install `git` separately.

Get the Flutter SDK

1- Download the following installation bundle to get the **3.16.4 Stable** release of the Flutter SDK:

```
$ cd ~/development$ unzip ~/Downloads/flutter_macos_3.16.4-stable.zip
```

2- Extract the file in the desired location, for example:

```
$ export PATH="$PATH:`pwd`/flutter/bin"
```

3- Add the `flutter` tool to your path:

This command sets your `PATH` variable for the *current* terminal window only. To permanently add Flutter to your path, see Update your path.

You are now ready to run Flutter commands!

Run flutter doctor

Run the following command to see if there are any dependencies you need to install to complete the setup (for verbose output, add the `-v` flag):

This command checks your environment and displays a report to the terminal window. The Dart SDK is bundled with Flutter; it is not necessary to install Dart separately. Check the output carefully for other software you might need to install or further tasks to perform (shown in **bold** text).

For example:

```
[-] Android toolchain - develop for Android devices    • Android SDK at /Users/obiwan/Librar
```

The following sections describe how to perform these tasks and finish the setup process.

Once you have installed any missing dependencies, run the `flutter doctor` command again to verify that you've set everything up correctly.

Downloading straight from GitHub instead of using an archive

This is only suggested for advanced use cases.

You can also use git directly instead of downloading the prepared archive. For example, to download the stable branch:

```
$ git clone https://github.com/flutter/flutter.git -b stable
```

Update your path, and run `flutter doctor`. That will let you know if there are other dependencies you need to install to use Flutter (e.g. the Android SDK).

If you did not use the archive, Flutter will download necessary development binaries as they are needed (if you used the archive, they are included in the download). You may wish to pre-download these development binaries (for example, you may wish to do this when setting up hermetic build environments, or if you only have intermittent network availability). To do so, run the following command:

For additional download options, see `flutter help precache`.

Update your path

You can update your PATH variable for the current session at the command line, as shown in *Get the Flutter SDK*. You'll probably want to update this variable permanently, so you can run `flutter` commands in any terminal session.

The steps for modifying this variable permanently for all terminal sessions are machine-specific. Typically you add a line to a file that is executed whenever you open a new window. For example:

1. Determine the path of your clone of the Flutter SDK. You need this in Step 3.
2. Open (or create) the `rc` file for your shell. Typing `echo $SHELL` in your Terminal tells you which shell you're using. If you're using Bash, edit `$HOME/.bash_profile` or `$HOME/.bashrc`. If you're using Z shell, edit `$HOME/.zshrc`. If you're using a different shell, the file path and filename will be different on your machine.
3. Add the following line and change `[PATH_OF_FLUTTER_GIT_DIRECTORY]` to be the path of your clone of the Flutter git repo:

   ```
   $ export PATH="$PATH:[PATH_OF_FLUTTER_GIT_DIRECTORY]/bin"
   ```
4. Run `source $HOME/.` to refresh the current window, or open a new terminal window to automatically source the file.
5. Verify that the `flutter/bin` directory is now in your PATH by running:
   Verify that the `flutter` command is available by running:

Platform setup

macOS supports developing Flutter apps in iOS, Android, and the web (technical preview release). Complete at least one of the platform setup steps now, to be able to build and run your first Flutter app.

iOS setup

Install Xcode

To develop Flutter apps for iOS, you need a Mac with Xcode installed.

1. Install the latest stable version of Xcode (using web download or the Mac App Store).
2. Configure the Xcode command-line tools to use the newly-installed version of Xcode by running the following from the command line:

   ```
   $ sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer$ sudo xcodebuild
   ```
   This is the correct path for most cases, when you want to use the latest version of Xcode. If you need to use a different version, specify that path instead.
3. Make sure the Xcode license agreement is signed by either opening Xcode once and confirming or running `sudo xcodebuild -license` from the command line.

Versions older than the latest stable version may still work, but are not recommended for Flutter development. Using old versions of Xcode to target bitcode is not supported, and is likely not to work.

With Xcode, you'll be able to run Flutter apps on an iOS device or on the simulator.

Set up the iOS simulator

To prepare to run and test your Flutter app on the iOS simulator, follow these steps:

1. On your Mac, find the Simulator via Spotlight or by using the following command:
2. Make sure your simulator is using a 64-bit device (iPhone 5s or later) by checking the settings in the simulator's **Hardware > Device** menu.
3. Depending on your development machine's screen size, simulated high-screen-density iOS devices might overflow your screen. Grab the corner of the simulator and drag it to change the scale. You can also use the **Window > Physical Size** or **Window > Pixel Accurate** options if your computer's resolution is high enough.
   - If you are using a version of Xcode older than 9.1, you should instead set the device scale in the **Window > Scale** menu.

Android setup

Install Android Studio

1. Download and install Android Studio.
2. Start Android Studio, and go through the 'Android Studio Setup Wizard'. This installs the latest Android SDK, Android SDK Command-line Tools, and Android SDK Build-Tools, which are required by Flutter when developing for Android.
3. Run `flutter doctor` to confirm that Flutter has located your installation of Android Studio. If Flutter cannot locate it, run `flutter config --android-studio-dir` to set the directory that Android Studio is installed to.

Set up your Android device

To prepare to run and test your Flutter app on an Android device, you need an Android device running Android 8 (API level 27) or higher.

1. Enable **Developer options** and **USB debugging** on your device. Detailed instructions are available in the Android documentation.
2. Windows-only: Install the Google USB Driver.
3. Using a USB cable, plug your phone into your computer. If prompted on your device, authorize your computer to access your device.
4. In the terminal, run the `flutter devices` command to verify that Flutter recognizes your connected Android device. By default, Flutter uses the version of the Android SDK where your `adb` tool is based. If you want Flutter to use a different installation of the Android SDK, you must set the `ANDROID_SDK_ROOT` environment variable to that installation directory.

Set up the Android emulator

To prepare to run and test your Flutter app on the Android emulator, follow these steps:

1. Enable VM acceleration on your machine.
2. Launch **Android Studio**, click the **AVD Manager** icon, and select **Create Virtual Device…**
   - In older versions of Android Studio, you should instead launch **Android Studio > Tools > Android > AVD Manager** and select **Create Virtual Device…**. (The **Android** submenu is only present when inside an Android project.)
   - If you do not have a project open, you can choose **Configure > AVD Manager** and select **Create Virtual Device…**
3. Choose a device definition and select **Next**.
4. Select one or more system images for the Android versions you want to emulate, and select **Next**. An *x86* or *x86_64* image is recommended.
5. Under Emulated Performance, select **Hardware - GLES 2.0** to enable hardware acceleration.
6. Verify the AVD configuration is correct, and select **Finish**.
   For details on the above steps, see Managing AVDs.
7. In Android Virtual Device Manager, click **Run** in the toolbar. The emulator starts up and displays the default canvas for your selected OS version and device.

Agree to Android Licenses

Before you can use Flutter, you must agree to the licenses of the Android SDK platform. This step should be done after you have installed the tools listed above.

1. Make sure that you have a version of Java 8 installed and that your `JAVA_HOME` environment variable is set to the JDK's folder.
   Android Studio versions 2.2 and higher come with a JDK, so this should already be done.
2. Open an elevated console window and run the following command to begin signing licenses.
   ```
   $ flutter doctor --android-licenses
   ```
3. Review the terms of each license carefully before agreeing to them.
4. Once you are done agreeing with licenses, run `flutter doctor` again to confirm that you are ready to use Flutter.

Connect Application to Server or Admin

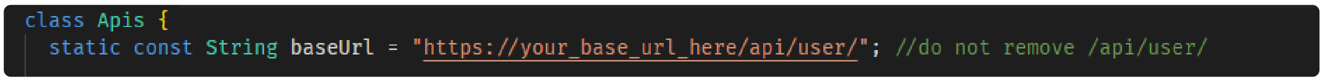
Make sure you have **HTTPS** enabled (secured) hosting.

User App

Go to Project Directory

Open File

`\lib\retrofit\apis.dart`



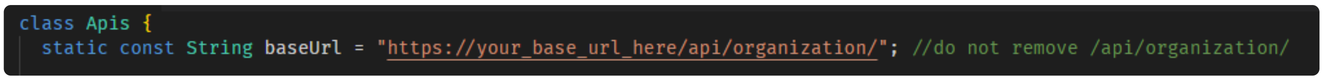
```
class Apis {
  static const String baseUrl = "https://your_base_url_here/api/user/"; //do not remove /api/user/
```

Organizer App

Go to Project Directory

Open File

`\lib\retrofit\apis.dart`



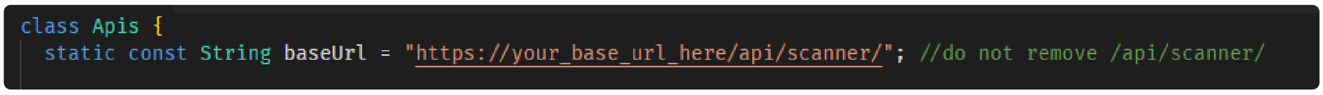
```
class Apis {
  static const String baseUrl = "https://your_base_url_here/api/organization/"; //do not remove /api/organization/
```

Scanner App

Go to Project Directory

Open File

`\lib\api\apis.dart`



```
class Apis {
  static const String baseUrl = "https://your_base_url_here/api/scanner/"; //do not remove /api/scanner/
```

Note: for some server directory, it may need `/public/` to be added before `/api`

- Execute the command :
  `flutter pub get && flutter pub run build_runner build --delete-conflicting-outputs`
  Then wait for the completion of the process.

Note: Everytime you modify the URL, you will have to re-run the command given above.

- While your app's source code is open in the IDE, & Enulator is Running
  Run the following command in the terminal:
  `flutter run`

Please note that as this is a URL(aka Route in Laravel Project), there's no need for the folder named `/api/user` to be there in the server directory.

Change Application name

Change Application name

For Android

- Go to below path and change the name **`android:label="Your_App_Name"`**
- File Location is : - **project/android/app/src/main/AndroidManifest.xml**.

```
<application
    android:label="Your_App_Name"        //Change this line
    android:usesCleartextTraffic="true"
    android:icon="@mipmap/ic_launcher">
```

For iOS

- Go to below path and change app name which seems like this **`<string>Your_App_Name</string>`**
- File Location is : - **project/ios/Runner/Info.plist.**

```
<key>CFBundleName</key>
<string>Your_App_Name</string>
```

Change Application package name (User, Guard and Owner)

Execute below commands in project/terminal.

1. **`flutter pub add change_app_package_name`**
2. **`flutter pub get`**
3. **`flutter pub run change_app_package_name:main com.new.package.name`**

   Note : - (replace your package name with this "com.new.package.name").

Change Bundle-identifier(User, Guard and Owner)

- Modify the bundle identifier from your `Info.plist` file inside your `ios/Runner` directory.

```
<key>CFBundleIdentifier</key>
<string>com.your.packagename</string> // change this line to your bundle-identifier
```

We can find `CFBundleIdentifier` used multiple places. Just search `CFBundleIdentifier`in your IDE and replace its value with our new bundle identifier.

Changing in-app Icon Image

These steps are for in-app logo image. Steps for splashscreen replacement are in next page

User App

1. Check `Icon.png` & `login_image.png` inside `assets/images/` folder.
2. Replace those images without changing the dimensions. Using Transparent PNG is recommanded for in-app images
3. After replacing those images, open project's terminal and run the following command
   \`flutter pub get\`
4. Rebuild the app. You will see those images changed.

Organizer App

1. Check `Icon.png` inside `assets/images/` folder.
2. Replace that image without changing the dimensions. Using Transparent PNG is recommanded for in-app images
3. After replacing those images, open project's terminal and run the following command
   \`flutter pub get\`
4. Rebuild the app. You will see those images changed.

Scanner App

1. Check `AppIcon.png` inside `assets/images/` folder.
2. Replace that image without changing the dimensions. Using Transparent PNG is recommanded for in-app images
3. After replacing those images, open project's terminal and run the following command
   \`flutter pub get\`
4. Rebuild the app. You will see those images changed.

Changing Launcher Icon

Step 1

Set your custom `image_path` in `pubspec.yaml` as follows:

```
flutter_icons:
  android: "launcher_icon"
  ios: true
  image_path: assets/images/icon.png
```

Step 2

Run the following commands from project's terminal, one by one.

```
flutter pub get
```

```
flutter pub run flutter_launcher_icons
```

Step 3:

Rebuild/Re-run the app.

Changing Splashscreen Image

These steps are bit complex to understand. Reading each step carefully is adviced.

Step 1

Replace the image path in the `pubspec.yaml` file in all three apps. You can change the RGB Hex color codes as well.

```
flutter_native_splash:
  color: "#2C64F4"
  image: "___Enter_your_splash_image_path___"
  color_dark: "#1a1a1a"
  image_dark: "___Enter_your_splash_image_path___"
  android: true
  ios: true
  web: true
  android12: true
  android_gravity: center
  ios_content_mode: center

  android_12:
   image: "___Enter_your_splash_image_path___"
   color: "#2C64F4"
   icon_background_color: "#2C64F4"
```

Following images are being used by default as of now.

User App

`assets/images/android_12.png`

Organizer App

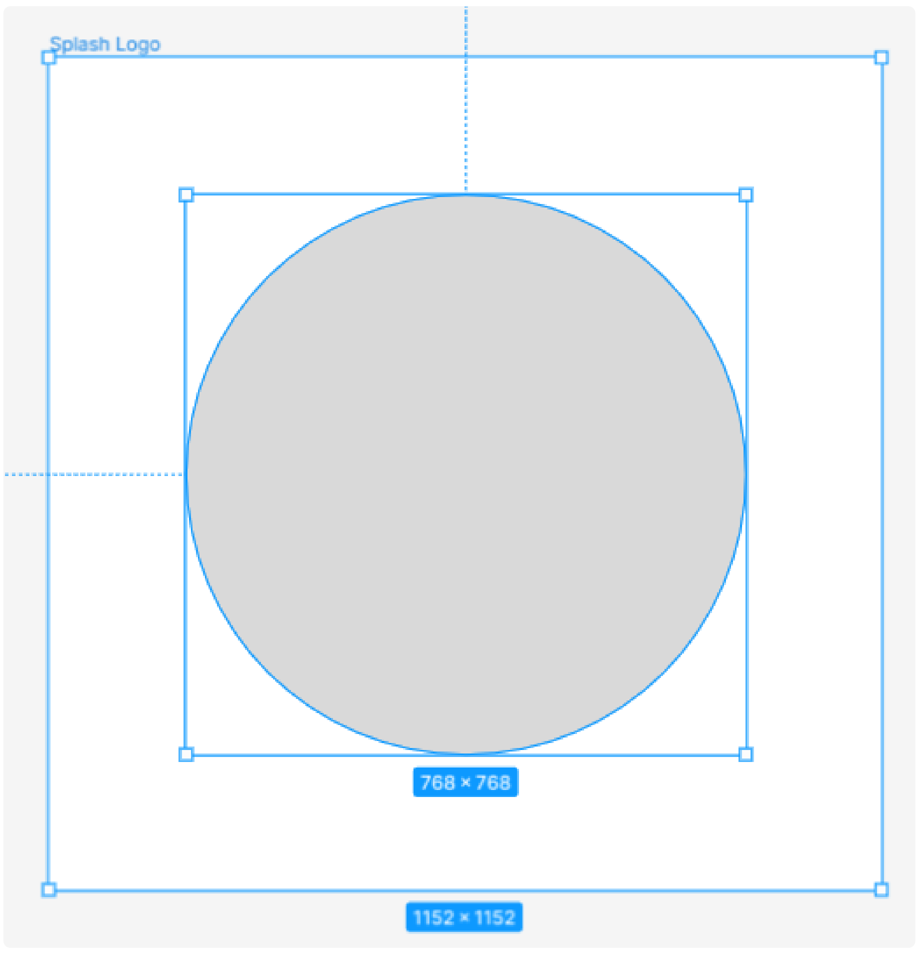
`assets/images/SplashScreen.png`

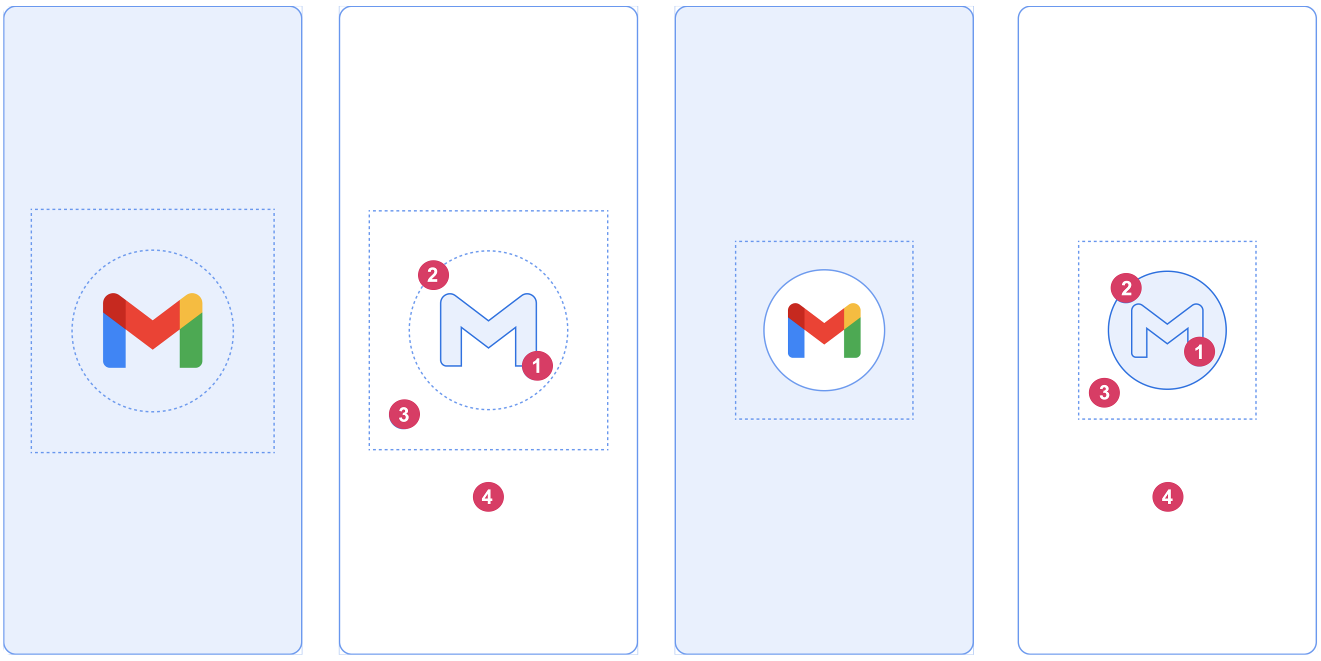
Scanner App

`assets/images/EventRight_Scanner_Spalsh12.png`

You should replace those images with your desired splashscreen logo image.

Please follow the below image guidelines to have properly set image. Our given splash images are already following this guidelines.





Be aware of the following considerations regarding these elements:

1. `image` parameter. By default, the launcher icon is used:
   - App icon without an icon background, as shown on the left: This should be 1152×1152 pixels, and fit within a circle 768 pixels in diameter.
   - App icon with an icon background, as shown on the right: This should be 960×960 pixels, and fit within a circle 640 pixels in diameter.
2. `icon_background_color` is optional, and is useful if you need more contrast between the icon and the window background.
3. One-third of the foreground is masked.
4. `color` the window background consists of a single opaque color.

PLEASE NOTE: The splash screen may not appear when you launch the app from Android Studio on API 31. However, it should appear when you launch by clicking on the launch icon in Android. This seems to be resolved in API 32+.

Step 2:

Run the following commands from project's terminal, one by one.

`flutter pub get`

`dart run flutter_native_splash:create`

Step 3:

Uninstall app in your device/emulator and re-run/re-build the app.
There are chances of splashimage not appearing on first debug run. But it'll surely be visible in the final build.

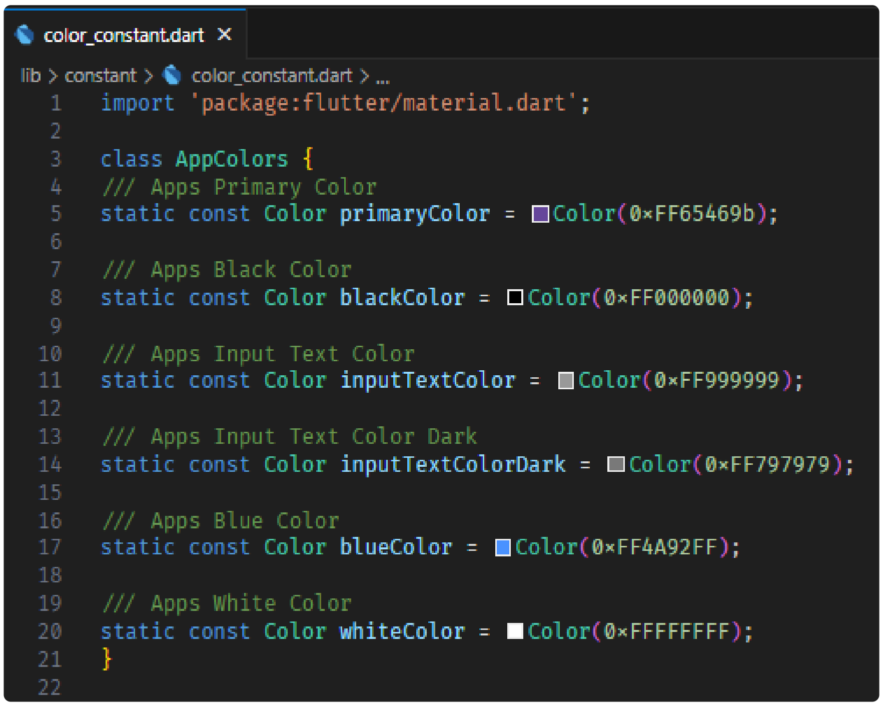
Changing the Colors

Color Format

0xFF**RRGGBB**
You should change only RRGGBB Hex part

User App

Modify the color values in `lib/constant/color_constant.dart`



color_constant.dart

lib > constant > color_constant.dart > ...

```dart
import 'package:flutter/material.dart';

class AppColors {
/// Apps Primary Color
static const Color primaryColor = Color(0xFF65469b);

/// Apps Black Color
static const Color blackColor = Color(0xFF000000);

/// Apps Input Text Color
static const Color inputTextColor = Color(0xFF999999);

/// Apps Input Text Color Dark
static const Color inputTextColorDark = Color(0xFF797979);

/// Apps Blue Color
static const Color blueColor = Color(0xFF4A92FF);

/// Apps White Color
static const Color whiteColor = Color(0xFFFFFFFF);
}
```

Localization (Optional)

These are the steps to add transaltions languages to the apps.

Caution: These are intermediate level code changes, make suree your code has a backup in case you make mistake.
Make sure you are following consistancy with casings.

User App & Organizer App:

Step: 1

First, create the JSON file of the language you want to add; in the following folder.

Folder::: lib->localization->languages.

Step 2

Add your language related code on following locations.

File::: lib->localization->localization_constant.dart.

Step 3:

Go to lib->main.dart

Add language code & country code

Step 4:

open following location
lib->localization->lanugaue.dart

Scanner App

Step 1:

First, create the JSON file of the language you want to add; in the following folder.

Folder::: lib->localization->languages.

Step 2

Add your language related code on following locations.

File::: lib->localization->localization_constant.dart.

Created language variable used in following switch case condition

Step 3:

Go to lib->main.dart

Add language code & country code

Step 4

go to following location set switch case statement proper value(your language code set)

Same as code structure

lib->Screen->Home->home_screen.dart

Step 5:

Go to following location

lib->Providers->setting_provider.dart
Using `else if` condition block, add your language condition

Run The App

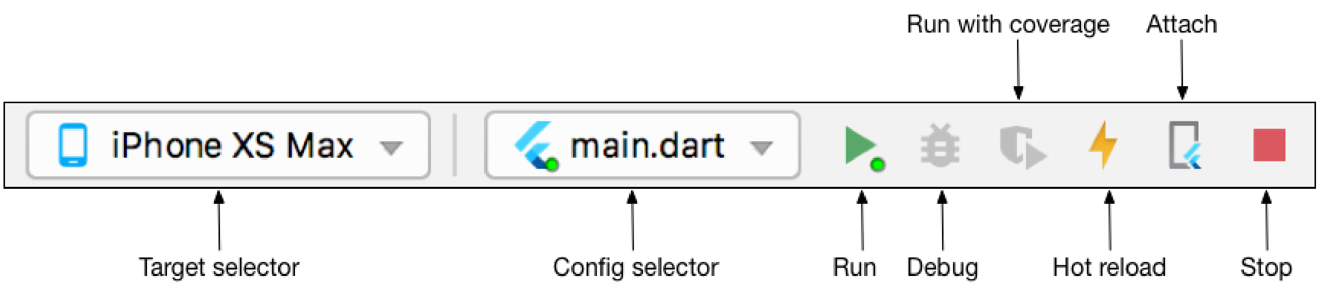
Using Android Studio

Open the app files

Select File from top list menu, and chose open folder then select the project folder.

Run the app

1- Locate the main Android Studio toolbar:



2- In the **target selector**, select an Android device for running the app. If none are listed as available, select **Tools> Android > AVD Manager** and create one there. For details, see Managing AVDs.

3- Click the run icon in the toolbar, or invoke the menu item **Run > Run**.

After the app build completes, you'll see the app on your device.

Starter app

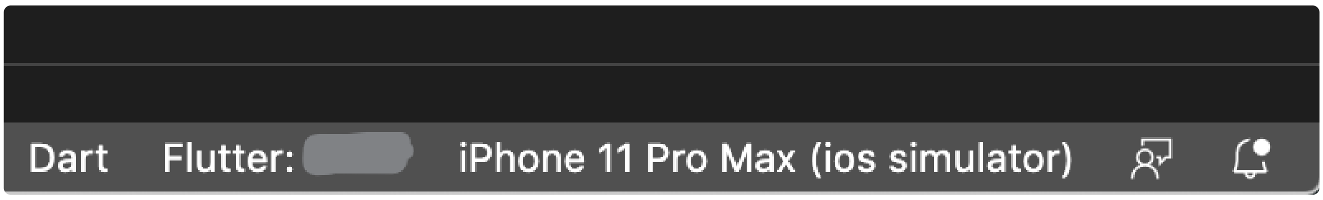
Using VS Code Studio

Open the app files

Select File from top list menu, and chose open folder then select the project folder.

Run the app

1- Locate the VS Code status bar (the blue bar at the bottom of the window):



2- Select a device from the **Device Selector** area. For details, see Quickly switching between Flutter devices.

- If no device is available and you want to use a device simulator, click **No Devices** and launch a simulator.

Warning: You may not see **Start iOS Simulator** option when you click **No Devices** in VS Code. If you are on Mac then you may have to run following command in terminal to launch a simulator.

In Android it is not possible to launch iOS simulator.

- To setup a real device, follow the device-specific instructions on the Install page for your OS.

3- Invoke **Run > Start Debugging** or press F5.

4- Wait for the app to launch — progress is printed in the **Debug Console** view.

After the app build completes, you’ll see the app on your device.

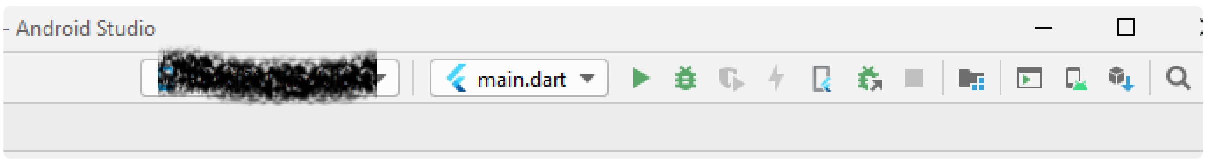
Note: Some functionality may not work on iOS Simulator. but it will work on a physical device.

Install App To Your Android Device

Install app to your android device

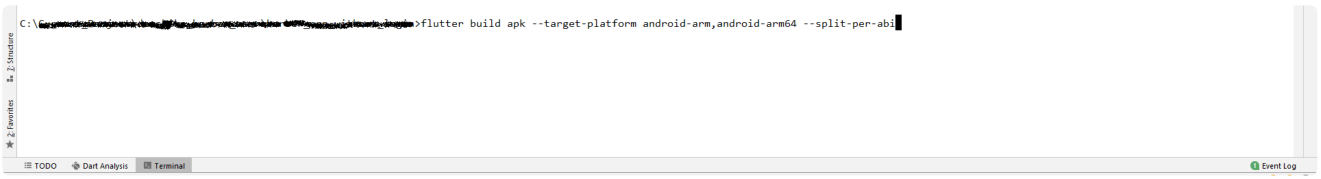
Steps for: How to install application in android device

1. Connect your android device via data transfer supported cable.
2. Open setting and turn on Developer option and turn on USB Debugging.
3. In android studio you can see your device connected at right hand side of screen.
4. then you need to just click on Green color play button to run it. it will take 3-6 minute as per your system configuration.



How to Generate APK file

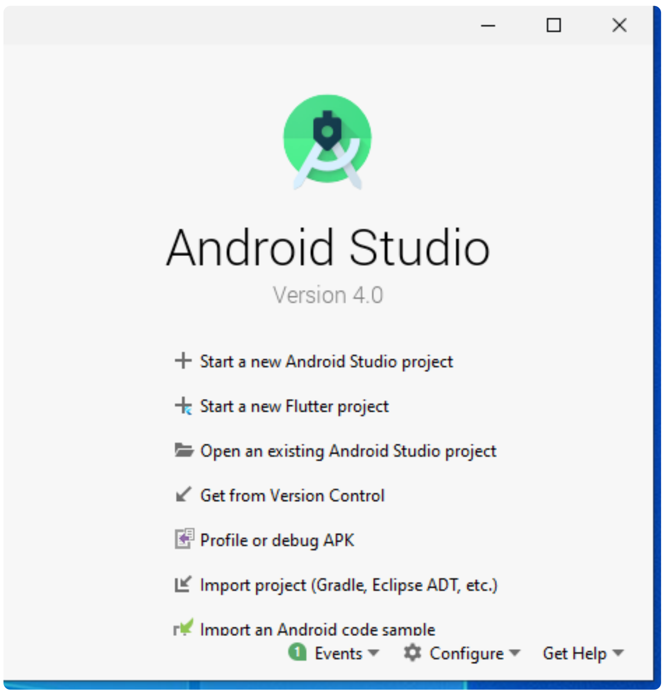
1. At bottom of android studio : TODO, Dart Analysis, and Terminal buttons available
2. Click on the Terminal button
3. execute command: `flutter build apk --target-platform android-arm,android-arm64 --split-per-abi`
4. after 3 to 6 minutes you can get APK file.



```
C:\[redacted]>flutter build apk --target-platform android-arm,android-arm64 --split-per-abi
```

How to Generate Signed APK

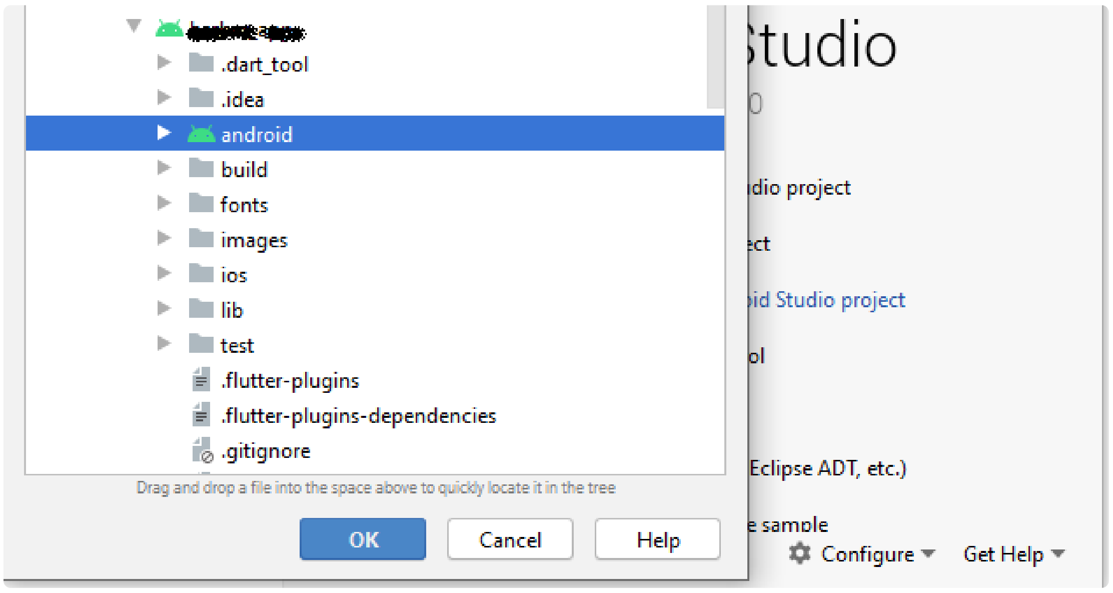
1. Open Android studio


Android Studio
Version 4.0
Start a new Android Studio project
Start a new Flutter project
Open an existing Android Studio project
Get from Version Control
Profile or debug APK
Import project (Gradle, Eclipse ADT, etc.)
Import an Android code sample
Events
Configure
Get Help

2. Click on "*Open an existing Android Studio project*"

3. Go to specific path of your code : [Project_Folder] -> android

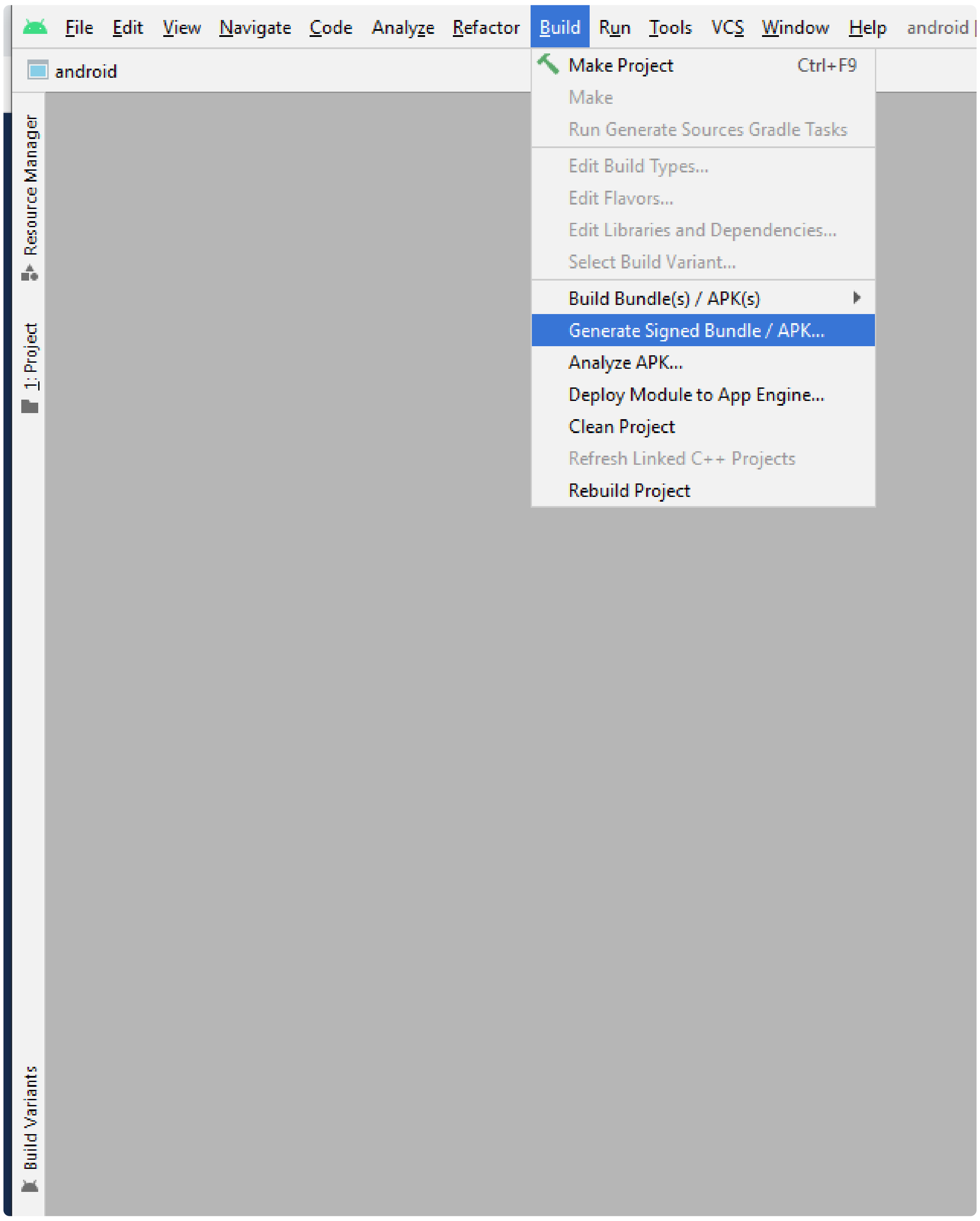
4. Select Android



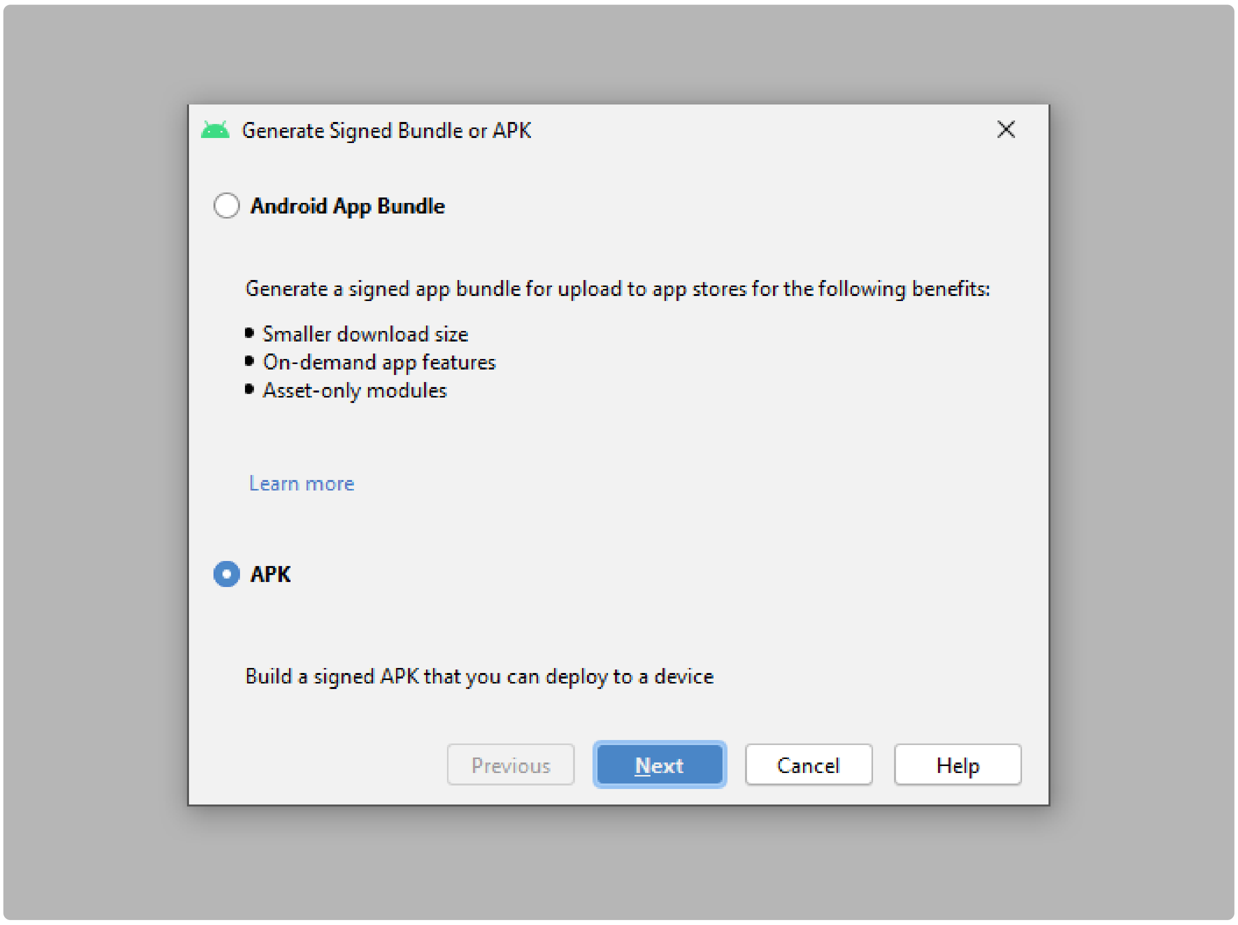
Click OK.

Wait while build process finishes successfully.

After that :



Select Build Menu -> Generate Signed Bundle/APK..


Generate Signed Bundle or APK

( ) **Android App Bundle**

Generate a signed app bundle for upload to app stores for the following benefits:

- Smaller download size
- On-demand app features
- Asset-only modules

Learn more

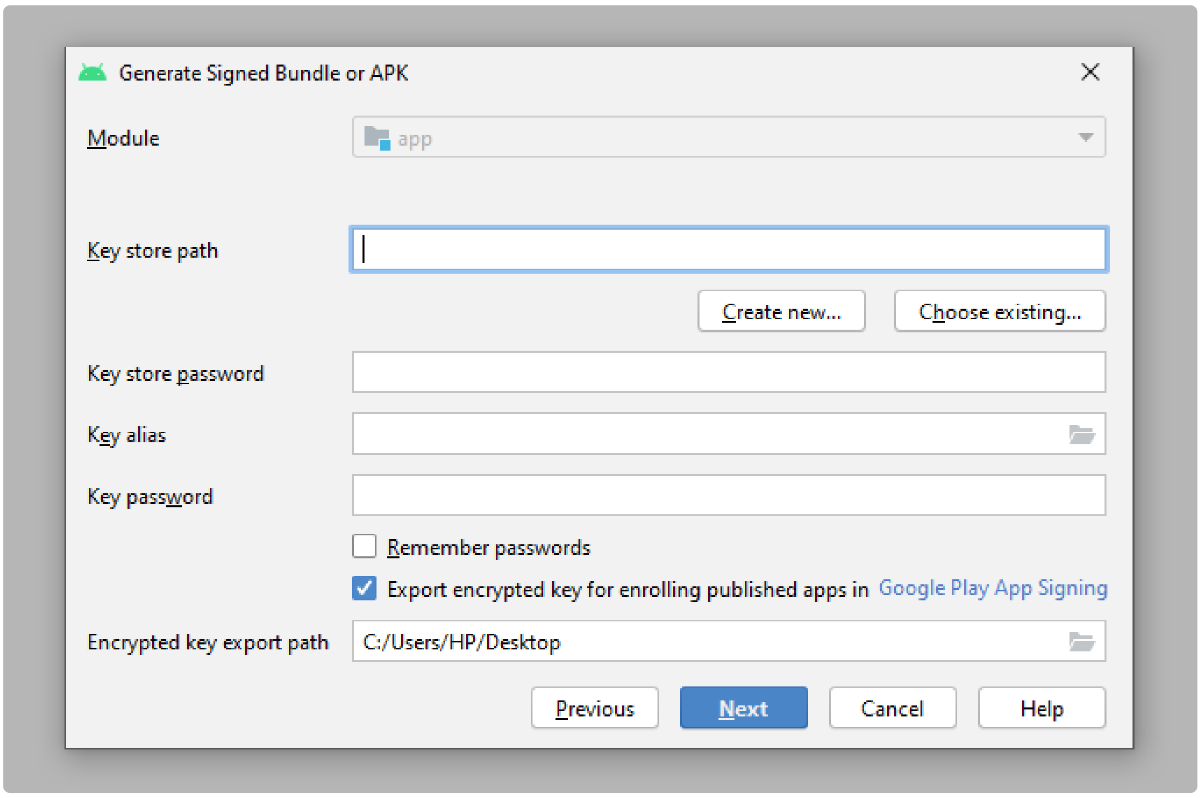
(●) **APK**

Build a signed APK that you can deploy to a device

Here you have 2 options:

- Android App Bundle
- APK

Select *Android App Bundle* to Generate Signed Bundle.



Generate Signed Bundle or APK

Module: app

Key store path:

Key store password:

Key alias:

Key password:

☐ Remember passwords

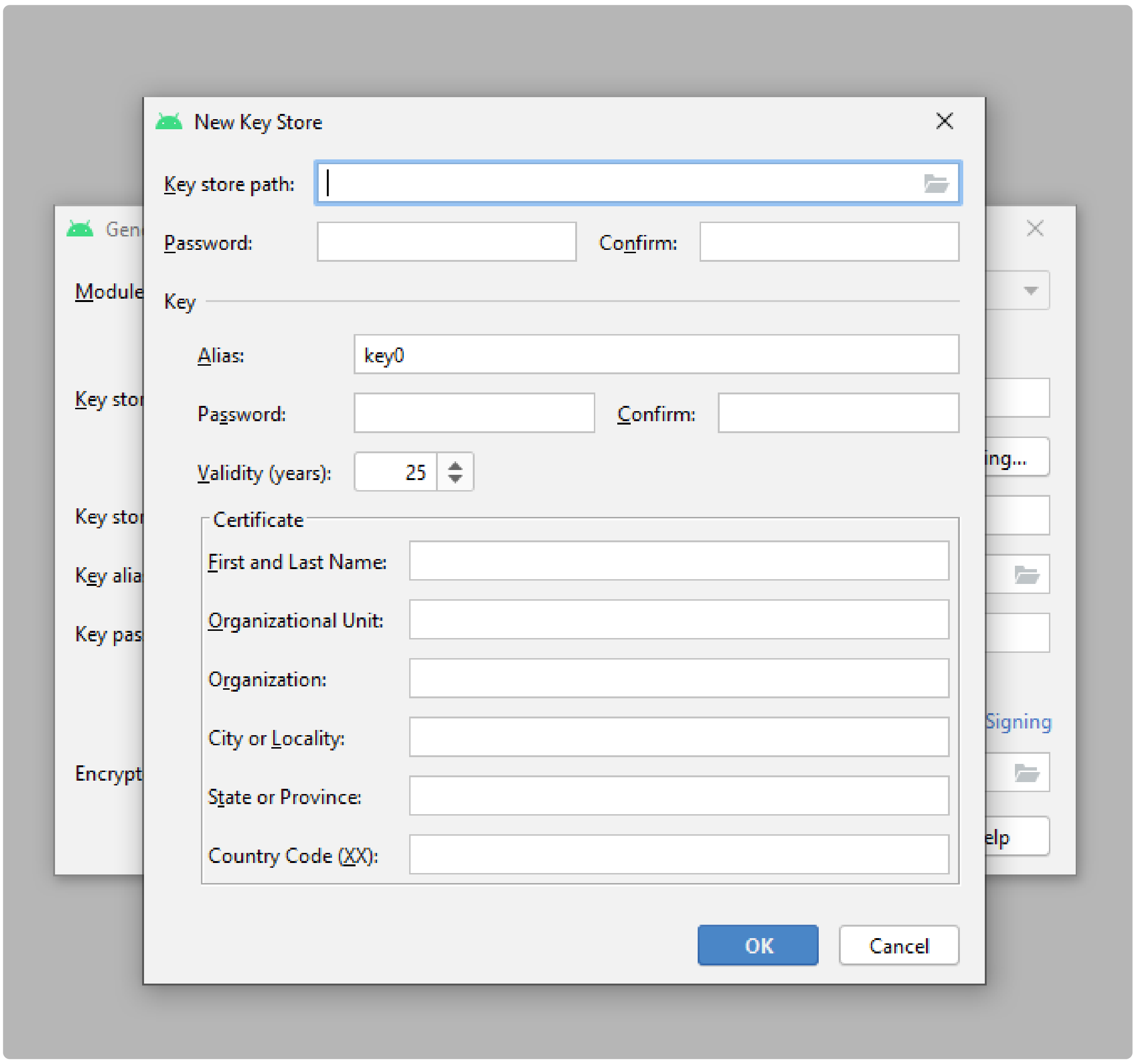
☑ Export encrypted key for enrolling published apps in Google Play App Signing

Encrypted key export path: C:/Users/HP/Desktop

You can see above dialog box :

If you are generating signed bundle/apk for first time, then you need to *CREATE NEW KEYSTORE PATH*

Click on "Create new..." You will see this dialog box


New Key Store

Key store path:

Password: Confirm:

Key

Alias: key0

Password: Confirm:

Validity (years): 25

Certificate

First and Last Name:

Organizational Unit:

Organization:

City or Locality:

State or Province:

Country Code (XX):

Fill-up all the necessary Details : Key store path , Key store Password, Key store confirm Password, KEY Alias, Alias Password, Alias confirm password, validity(years), Certificate: First & Last Name

Other details are optional.

Select key store path :

NOTE : (Select a specific path where you want to store KEY STORE File. This file will use while every time when you want to publish apk or Update application to Google Play store. So, Save it on safe location)

Set Password and Confirm Password

Key-----------

Alias: Set Alias Name

Alias Password: Set Alias Password

Alias Confirm Password: Set Alias Confirm Password

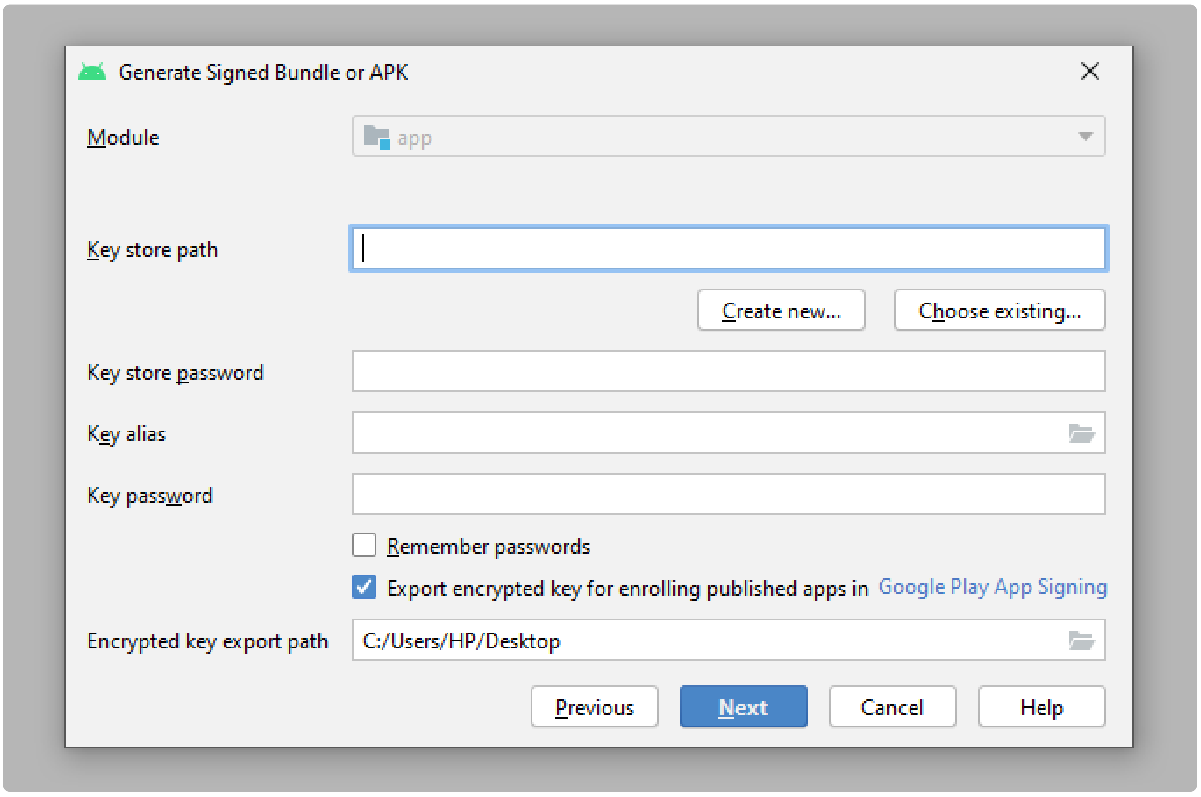
Validity : in years

Certificate :

First name and Last Name:

AFTER Filling up all the details, Click OK

Then you can see previous dialog box with key store path filled as your given data.


Generate Signed Bundle or APK

Module: app

Key store path:

Key store password:

Key alias:

Key password:

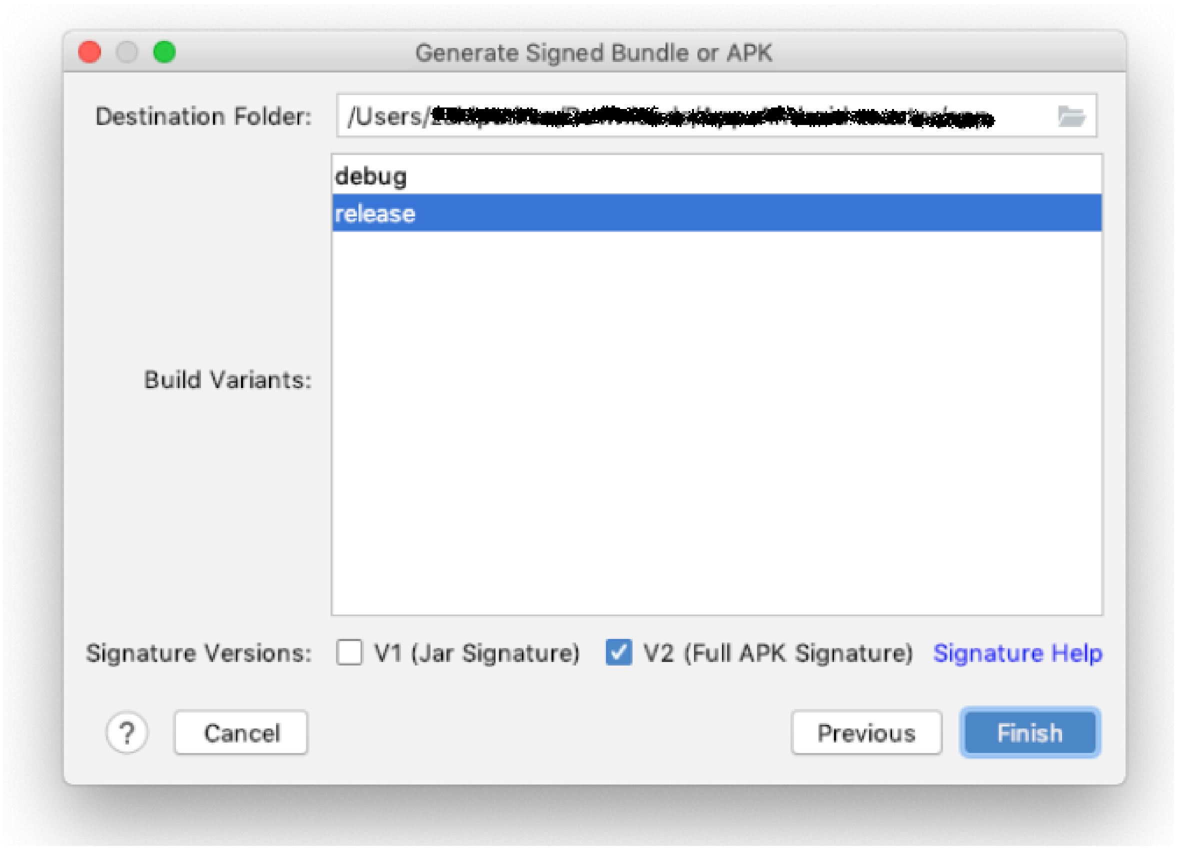
☐ Remember passwords

☑ Export encrypted key for enrolling published apps in Google Play App Signing

Encrypted key export path: C:/Users/HP/Desktop

Enter Keystore password , key alias , and key Password as you given while creating keystore path.

Click NEXT


Generate Signed Bundle or APK

Destination Folder: /Users/[redacted]

Build Variants:
- debug
- release

Signature Versions: [ ] V1 (Jar Signature) [✓] V2 (Full APK Signature) Signature Help

Here you need to select build Variants : release

Signature versions : V1 and V2

Select V2 Signature version for secure apk file from reverse engineering.

Click Finish.

Generating Signed apk will take 3-6 minute.

How to publish APK to Google play store

How to publish APK to Google play store

Here are steps for Android application publishing on the Google play store:

To publish your application on the google play store you need to have a Developer Account on Google Play

Create a Developer account using the below link:

https://play.google.com/apps/publish

The First Step to publish APK to google play store, We need to generate Signed APK. (Regular Build APK can not be published on the google play store.)

If you have knowledge of how to generate a signed APK then you can proceed to the next page.

Upload APK/App bundle to Google play store

After Successfully Generation of Signed APK, you can publish it on Google play store.

As pervious mentioned you must have Google developer / google play store account to complete the process of it. Create and set up your app - Play Console Help Starting August 2021, new apps will be required to publish with the Android App Bundle on Google Play.

New apps larger than 150MB can use either Play Asset Delivery or Play Feature Delivery. Reasupport.google.com

Thankyou

How to publish an app to appstore

Releasing Flutter iOS app to TestFlight using XCode

In this tutorial, we will be releasing Flutter iOS application on TestFlight using XCode. This is a beginner's guide for iOS release. This tutorial is intended for first time users who want to release apps built using Flutter to App Store.

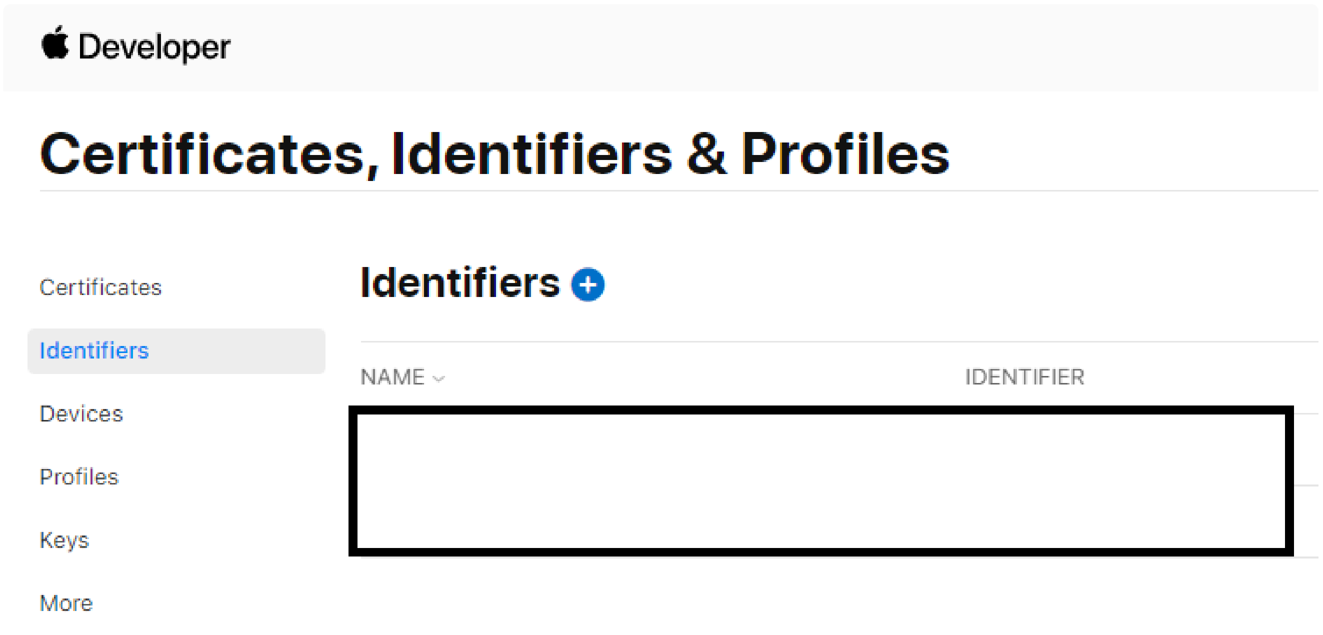
Flutter AppCenter TestFlight

To follow along with this tutorial, you need an Apple developers account, Flutter application and a MacOS device with XCode.

Create new App Bundle Identifier

Navigate to https://developer.apple.com/account and login to your Apple Developers account.

Go to Identifiers and create a new ID. This bundle ID is required to create app in App Store.



Developer

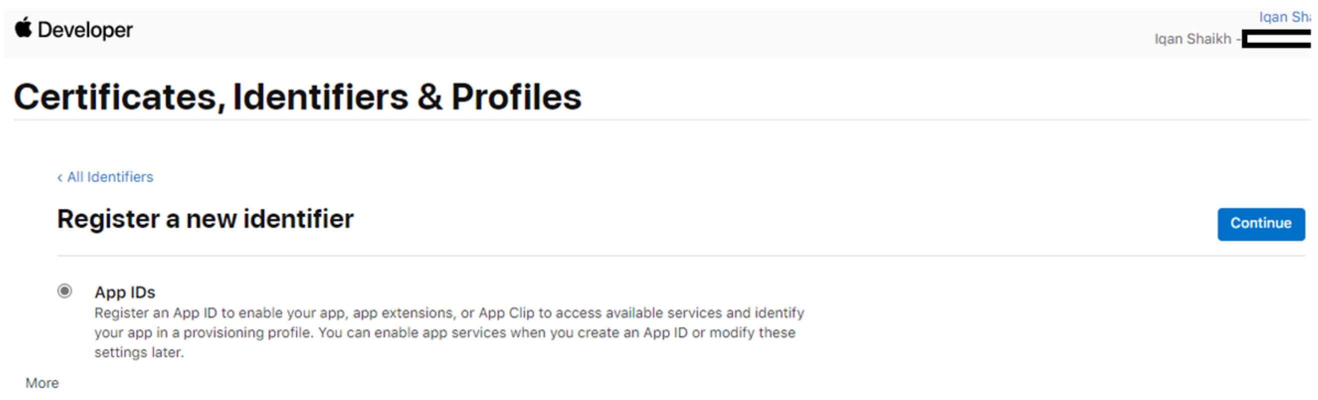
Certificates, Identifiers & Profiles

- Certificates
- Identifiers
- Devices
- Profiles
- Keys
- More

Identifiers

NAME	IDENTIFIER
[redacted]	

Select App IDs and continue



Developer

Iqan Sh
Iqan Shaikh [redacted]

Certificates, Identifiers & Profiles

‹ All Identifiers

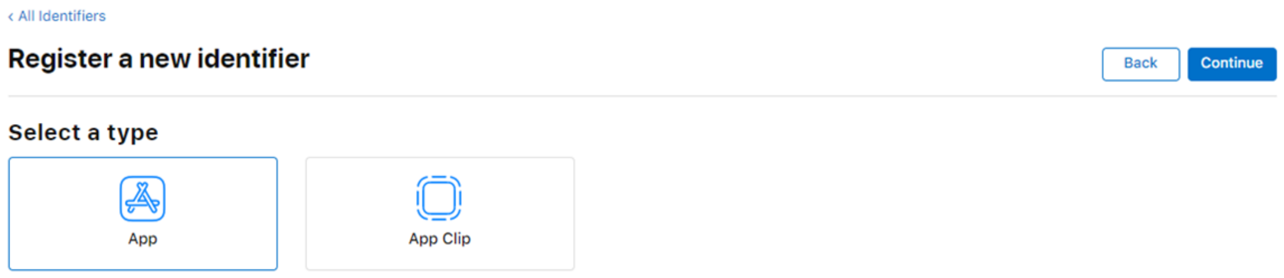
Register a new identifier

Continue

● App IDs
Register an App ID to enable your app, app extensions, or App Clip to access available services and identify your app in a provisioning profile. You can enable app services when you create an App ID or modify these settings later.

More

Select App to Register



‹ All Identifiers

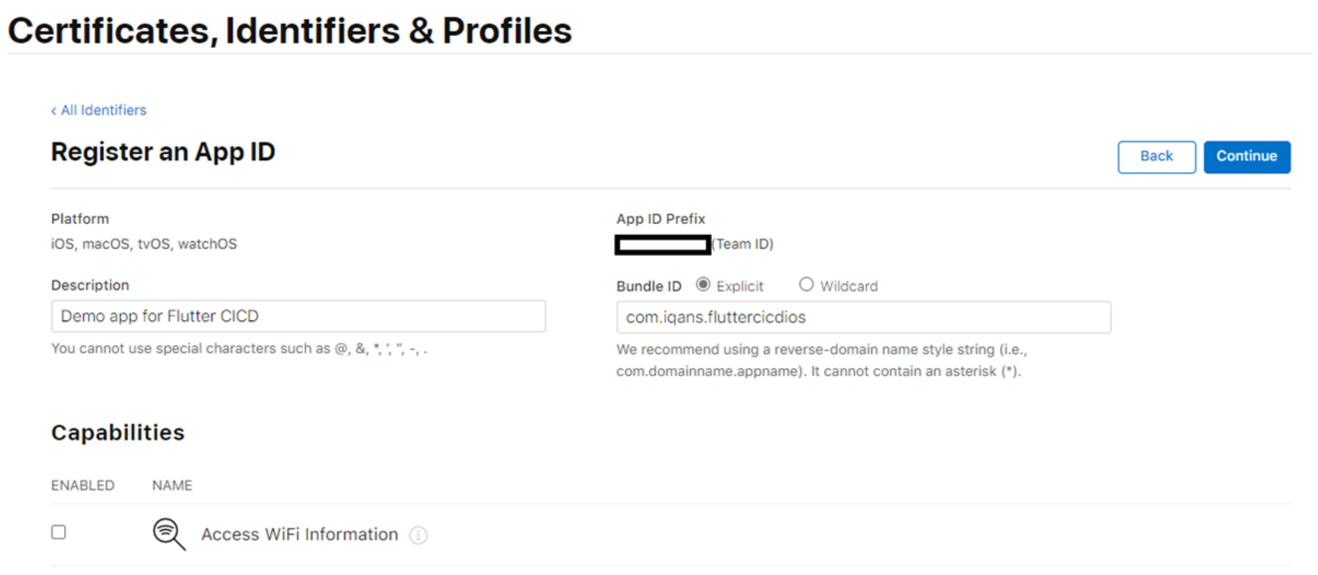
Register a new identifier

Back Continue

Select a type

[✓ App] App Clip

Provide description, bundle ID and check capabilities you need in you app



Certificates, Identifiers & Profiles

‹ All Identifiers

Register an App ID

Platform
iOS, macOS, tvOS, watchOS

App ID Prefix
[redacted] (Team ID)

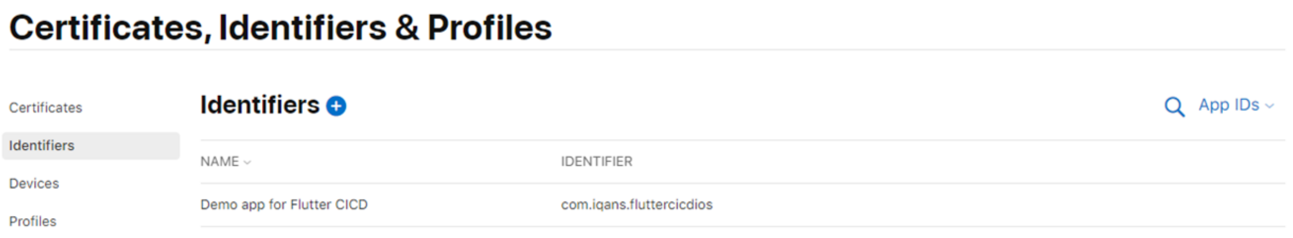
Description
Demo app for Flutter CICD
You cannot use special characters such as @, &, *, ', ", -, .

Bundle ID [✓ Explicit] Wildcard
com.iqans.fluttercicdios
We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

Capabilities

ENABLED	NAME
☐	Access WiFi Information

Once done, you will have an Identifier created for the app.



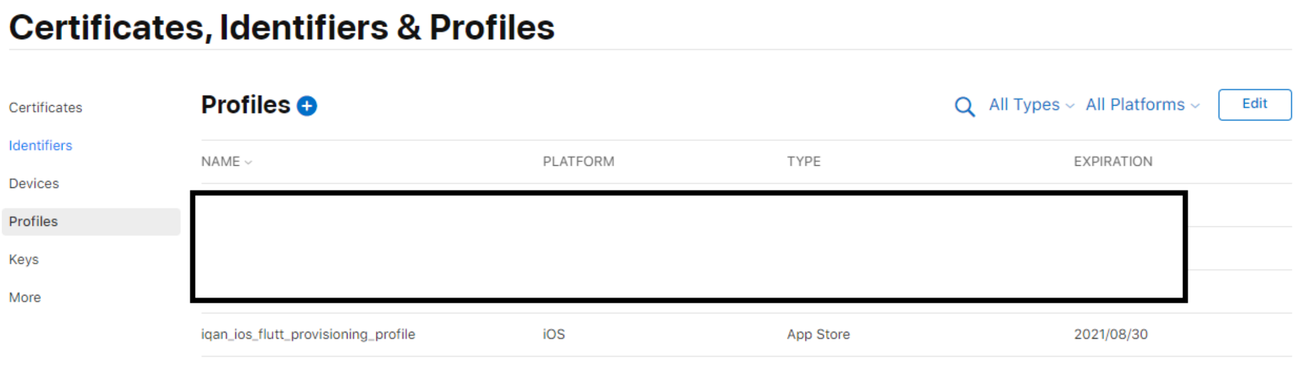
Certificates, Identifiers & Profiles

Identifiers

NAME	IDENTIFIER
Demo app for Flutter CICD	com.iqans.fluttercicdios

Create new provisioning Profile

Go to Profiles section and create new profile for provisioning

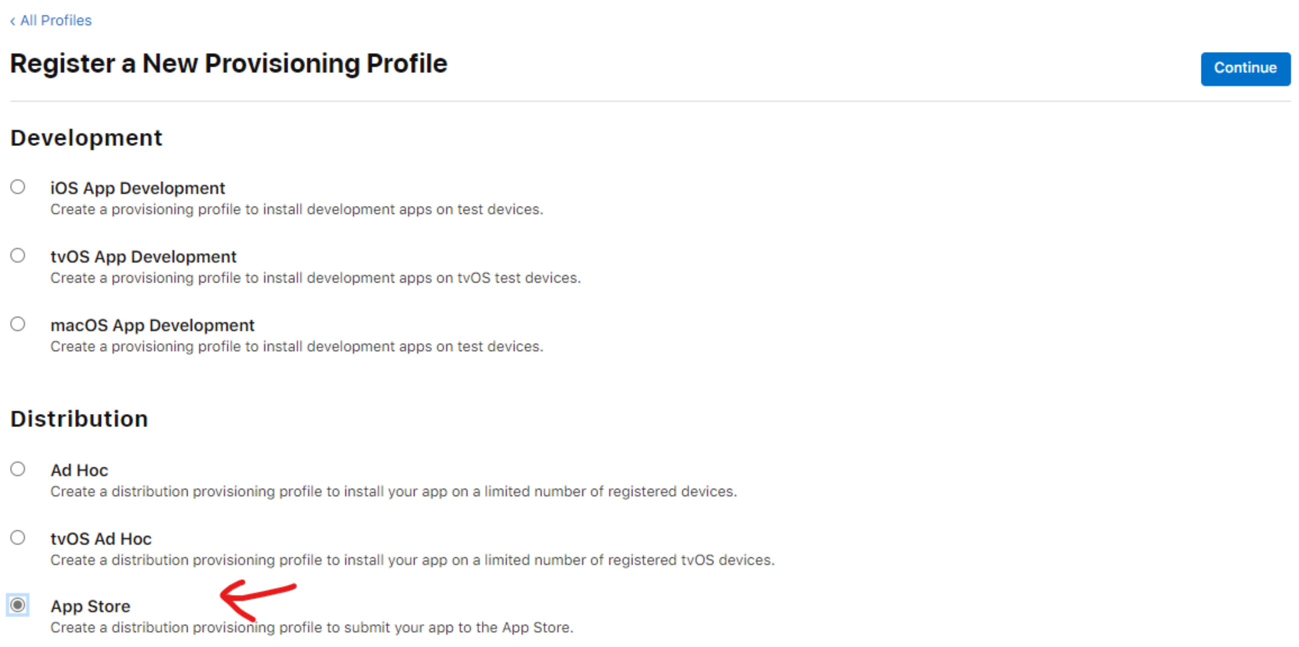


Certificates, Identifiers & Profiles

Profiles

NAME	PLATFORM	TYPE	EXPIRATION
[redacted]			
iqan_ios_flutt_provisioning_profile	iOS	App Store	2021/08/30

Select Distribution > App Store


‹ All Profiles

Register a New Provisioning Profile

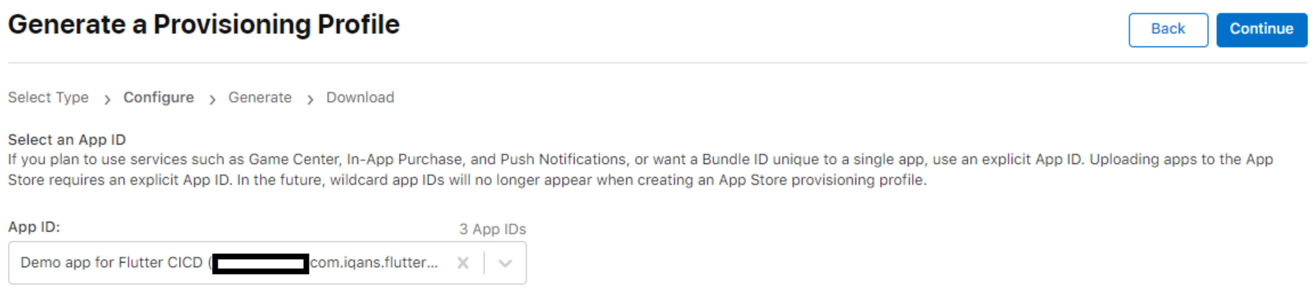
Development

- ○ iOS App Development — Create a provisioning profile to install development apps on test devices.
- ○ tvOS App Development — Create a provisioning profile to install development apps on tvOS test devices.
- ○ macOS App Development — Create a provisioning profile to install development apps on test devices.

Distribution

- ○ Ad Hoc — Create a distribution provisioning profile to install your app on a limited number of registered devices.
- ○ tvOS Ad Hoc — Create a distribution provisioning profile to install your app on a limited number of registered tvOS devices.
- [✓ App Store] — Create a distribution provisioning profile to submit your app to the App Store.

Select App ID and continue, this will create a new provisioning profile.

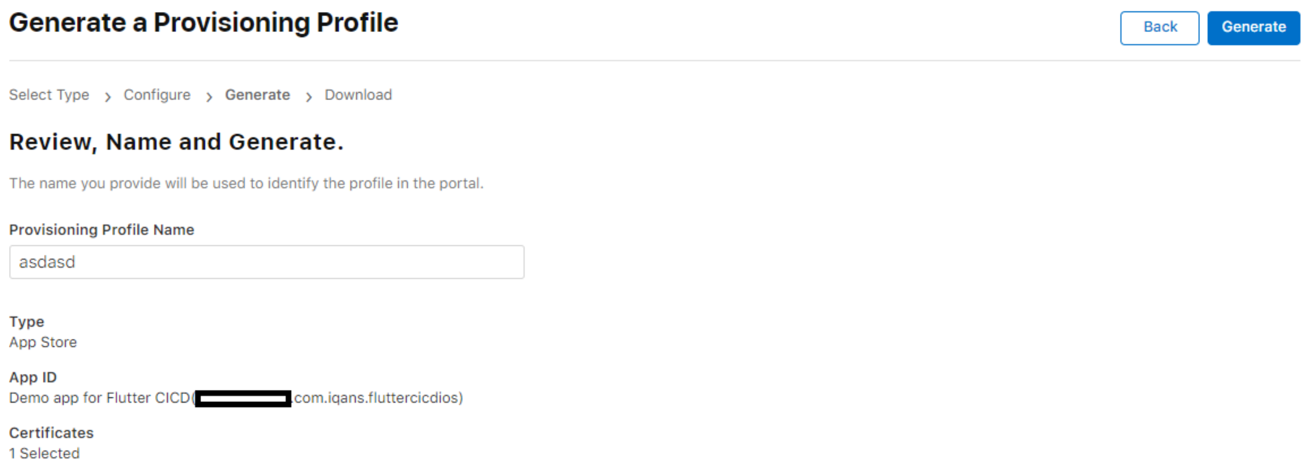

Generate a Provisioning Profile

Select Type › **Configure** › Generate › Download

Select an App ID
If you plan to use services such as Game Center, In-App Purchase, and Push Notifications, or want a Bundle ID unique to a single app, use an explicit App ID. Uploading apps to the App Store requires an explicit App ID. In the future, wildcard app IDs will no longer appear when creating an App Store provisioning profile.

App ID: 3 App IDs

Demo app for Flutter CICD ([redacted]com.iqans.flutter...


Generate a Provisioning Profile

Select Type › Configure › **Generate** › Download

Review, Name and Generate.

The name you provide will be used to identify the profile in the portal.

Provisioning Profile Name
asdasd

Type
App Store

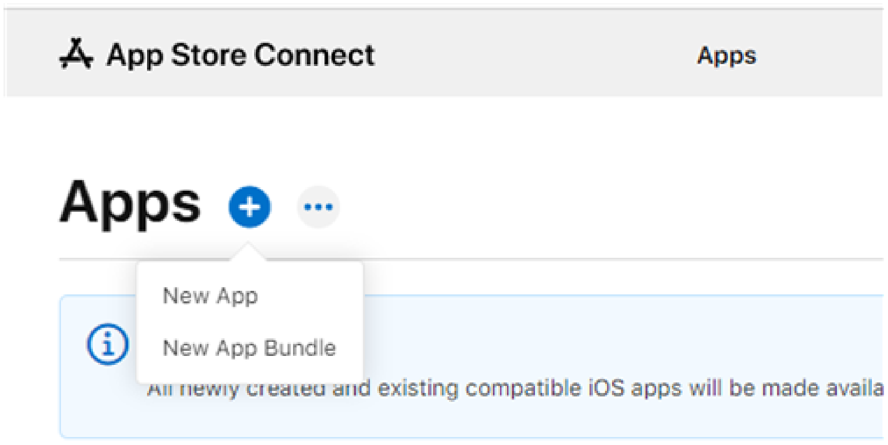
App ID
Demo app for Flutter CICD([redacted]com.iqans.fluttercicdios)

Certificates
1 Selected

Now, download that profile in local machine. This will be used in publishing app to Test Flight.

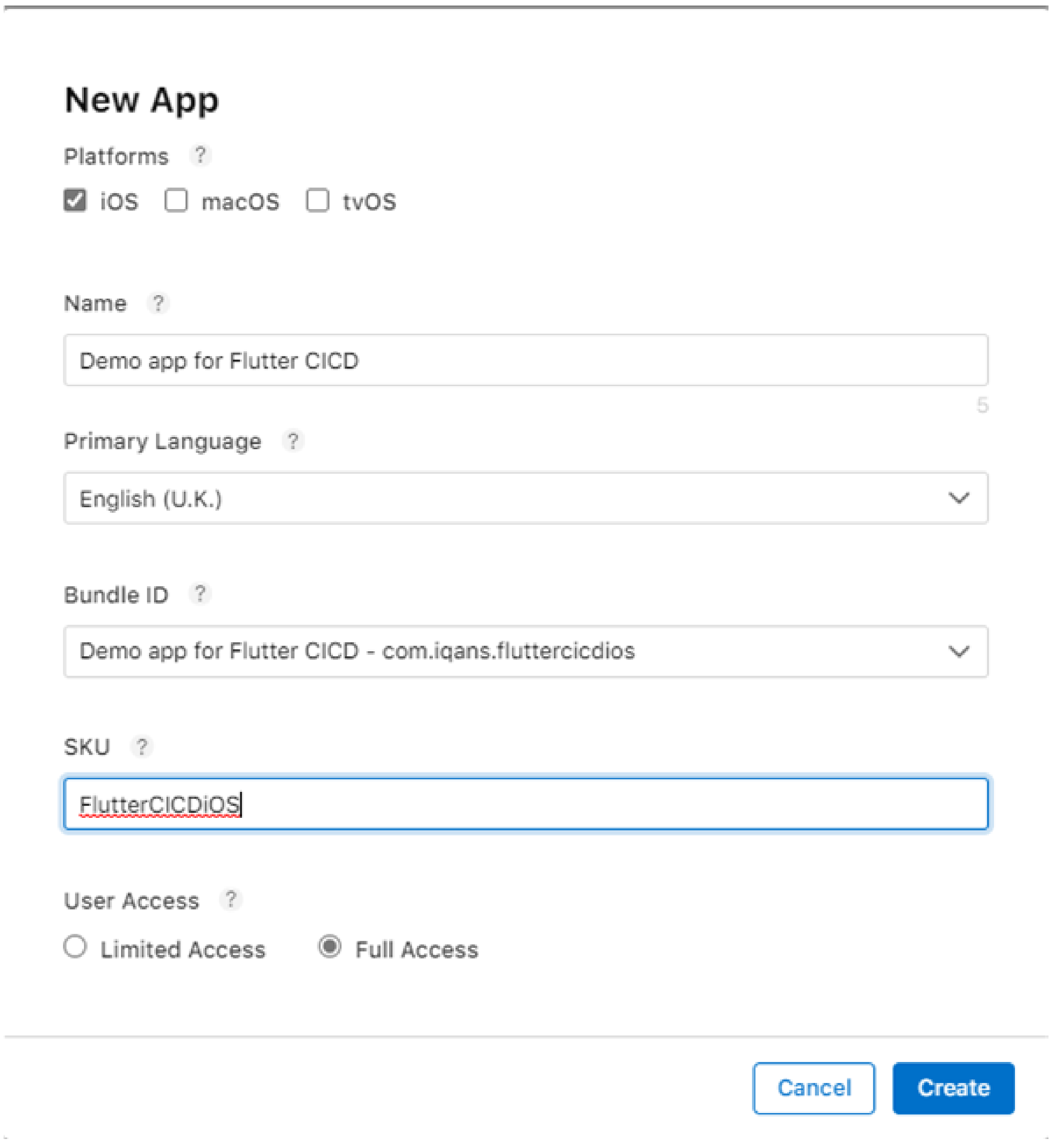
Create a new application in App Store

Navigate to App Store connect https://appstoreconnect.apple.com/apps and login. Go to Apps and create a New App



All newly created and existing compatible iOS apps will be made availa

Fill required details in the form and click on Create


New App

Platforms ?

[✓ iOS] macOS tvOS

Name ?

Demo app for Flutter CICD

5

Primary Language ?

English (U.K.)

Bundle ID ?

Demo app for Flutter CICD - com.iqans.fluttercicdios

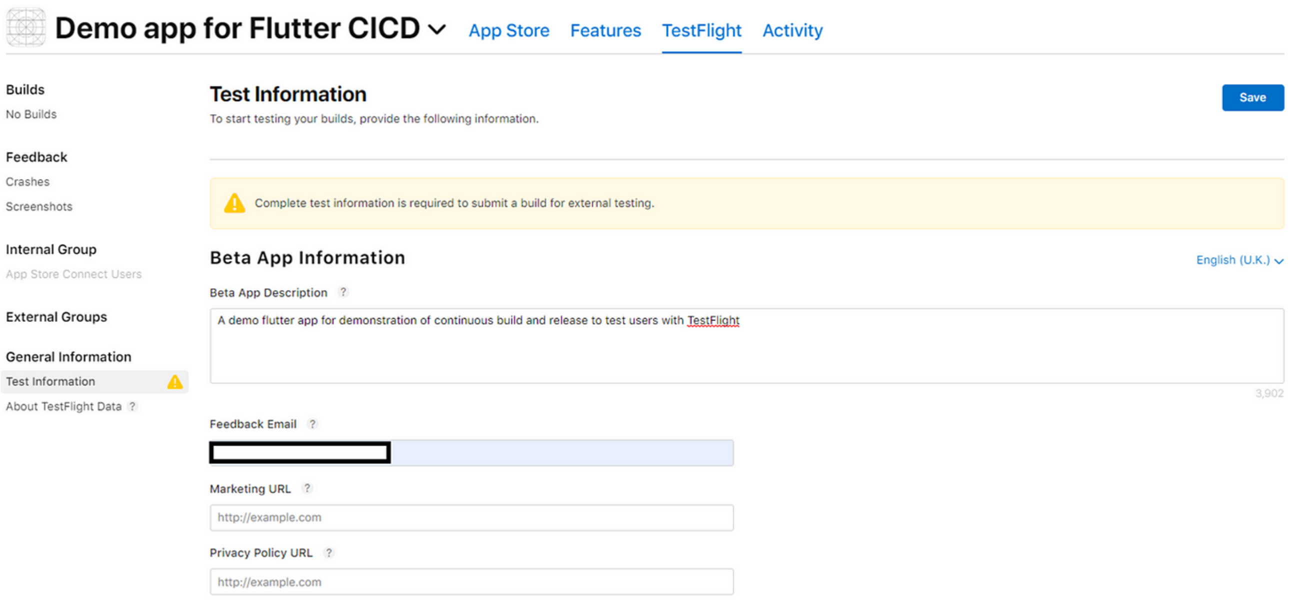
SKU ?

FlutterCICDiOS

User Access ?

Limited Access [✓ Full Access]

Once done, you will have an app created. Now go to TestFlight and fill mandatory details


Demo app for Flutter CICD

App Store Features TestFlight Activity

Builds
No Builds

Feedback
Crashes
Screenshots

Internal Group
App Store Connect Users

External Groups

General Information
Test Information
About TestFlight Data

Test Information

To start testing your builds, provide the following information.

Save

Complete test information is required to submit a build for external testing.

Beta App Information

English (U.K.)

Beta App Description

A demo flutter app for demonstration of continuous build and release to test users with TestFlight

3,902

Feedback Email

[redacted]

Marketing URL

http://example.com

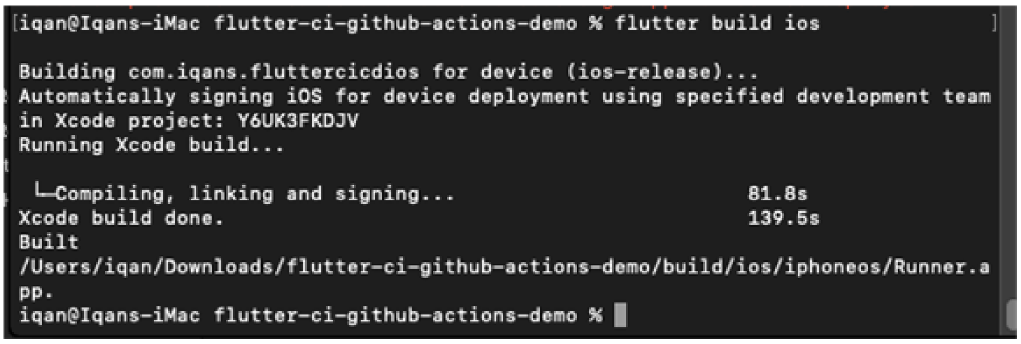
Privacy Policy URL

http://example.com

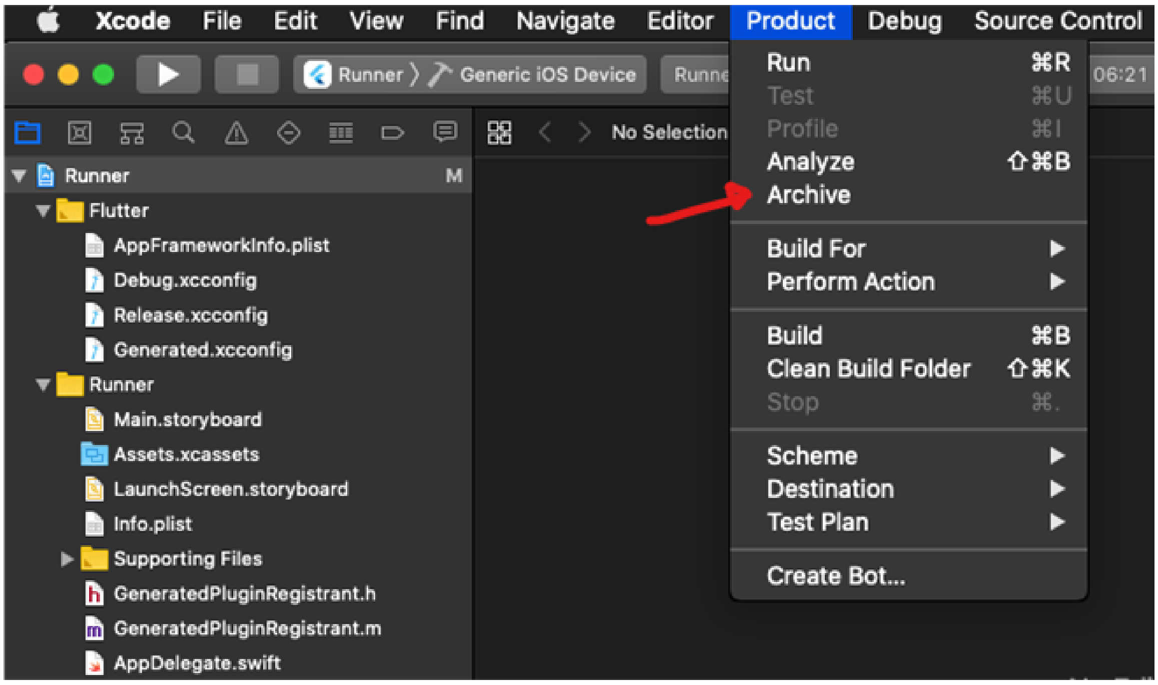
Submitting the app to TestFlight for the First time

App Center allows only updating existing apps. In order to use automated release process, you will need to upload your app to TestFlight from Xcode only once.

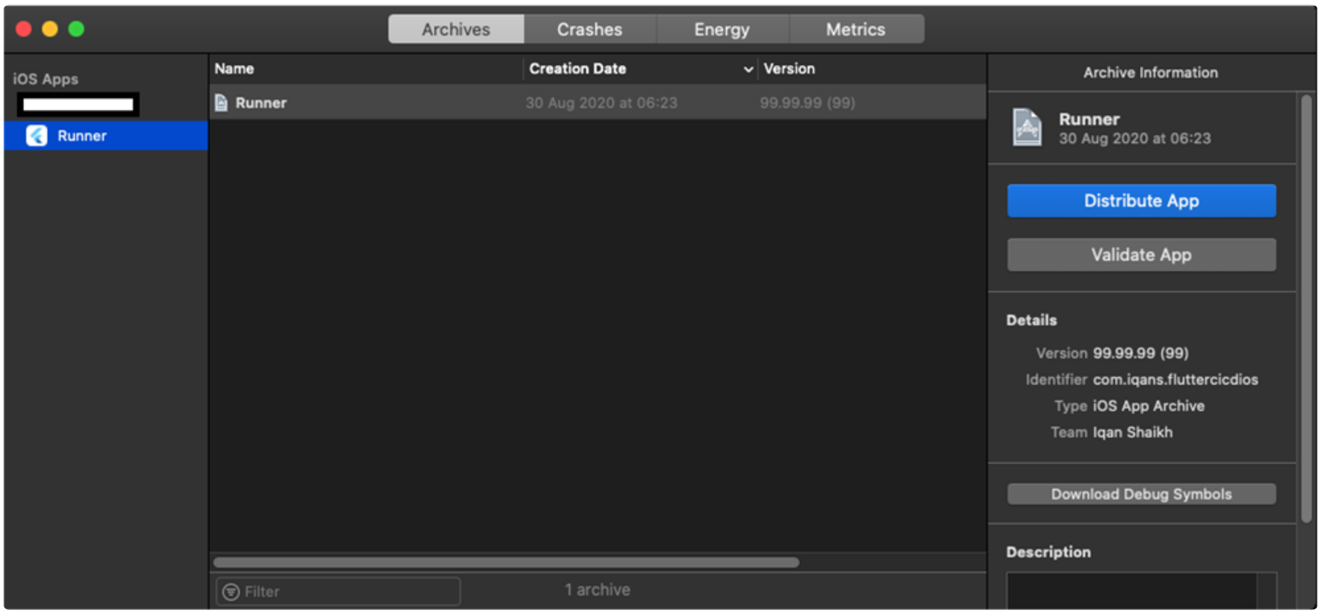
Build your flutter app for iOS.


```
iqan@Iqans-iMac flutter-ci-github-actions-demo % flutter build ios

Building com.iqans.fluttercicdios for device (ios-release)...
Automatically signing iOS for device deployment using specified development team
in Xcode project: Y6UK3FKDJV
Running Xcode build...

 └─Compiling, linking and signing...                          81.8s
Xcode build done.                                             139.5s
Built
/Users/iqan/Downloads/flutter-ci-github-actions-demo/build/ios/iphoneos/Runner.a
pp.
iqan@Iqans-iMac flutter-ci-github-actions-demo %
```

Open your app in Xcode. Go to Product > Archive



Once the archive is created, you can distribute app to app store in TestFlight



Name	Creation Date	Version
Runner	30 Aug 2020 at 06:23	99.99.99 (99)

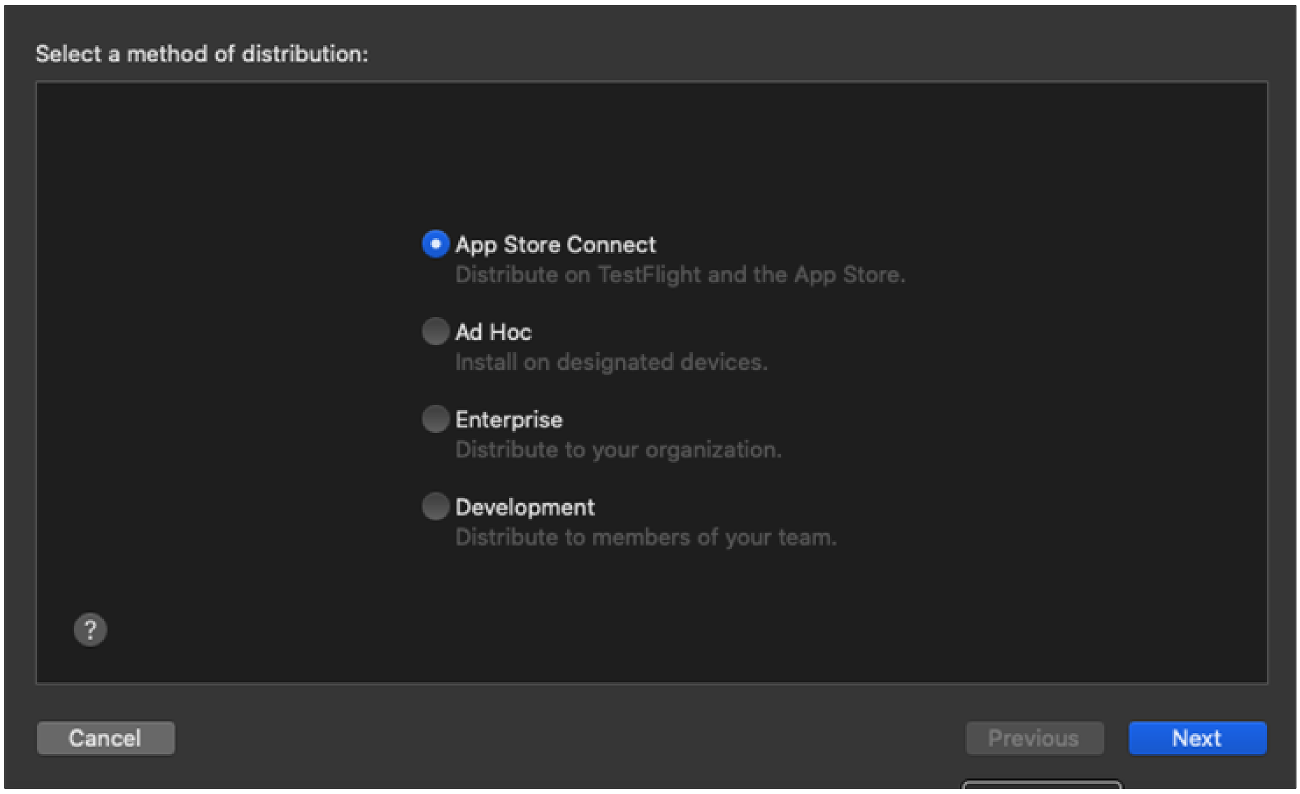
Runner
30 Aug 2020 at 06:23

Details

Version	99.99.99 (99)
Identifier	com.iqans.fluttercicdios
Type	iOS App Archive
Team	Iqan Shaikh

Description

Select App Store Connect as distribution method



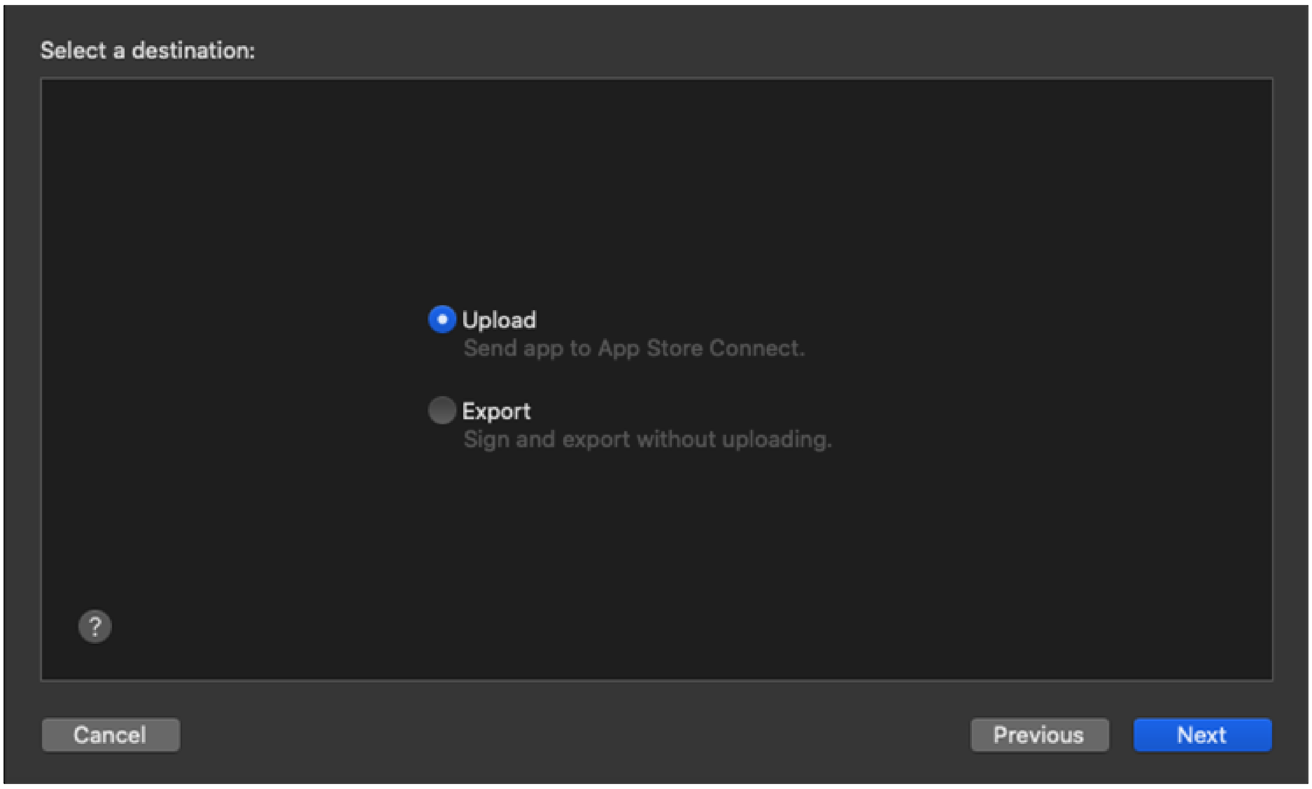
Select a method of distribution:

[✓ App Store Connect]
Distribute on TestFlight and the App Store.

Ad Hoc
Install on designated devices.

Enterprise
Distribute to your organization.

Development
Distribute to members of your team.



Select a destination:

[✓ Upload]
Send app to App Store Connect.

Export
Sign and export without uploading.

Here, you can create a new Certificate to sign your app.

> If you do that, please export that certificate with private key to a .p12 file. This is needed in App Center to sign your apps.

Thank you

End of Documentation

Thank you for purchasing our product

We appreciate your Purchase again.

For server/hosting issues or queries please contact your hosting provider support instead.