

Doctro - On Demand Doctor Appointment Booking System

Welcome to Doctro

Doctro+
On-Demand Doctor Appointment Booking
SaaS Business Model

- Multiple Payment Gateway
- 4 Apps In One Package
- SaaS Subscription Business Model
- User App + Doctor App Addon*

Top 5 Most Used Payment Gateways

PayPal, Razorpay, stripe, Flutterwave, paystack

Website + Doctor Panel + Pharmacy Panel + Admin Panel

Admin Panel

Doctor Panel
Doctor Dashboard

Pharmacy
Pharmacy Dashboard

Website

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

Admin Panel Setup

Server requirements

let's jump to the basic requirement of the server. any server with Linux installed will work but for better results and good performance, we recommend using a good VPS server with 2 GB or more ram.

PHP Version & Extension Required

- [PHP](#) version 7.3 and Above
- MySQLi PHP extension available

Doctro Installation

Go back to your cPanel home and **click on “File Manager.”** It will open in a new tab. Go to the directory you want the script. For your primary domain root directory, you will have to go to “**public_html**” click on **Upload** and **upload the PHP Script into the directory.**

The script you uploaded is likely to be in a **zip file**. To unzip it, simply right click on the zip file and click on **Extract**. The script will be in files or/and folders.

Before start installation process first of all check .env file is exist in your root directory in your uploaded script. If you do not find .env file, you may have to contact [Support](#).

Run Installer

Open your URL: **https://YourURL**

Let's start the installation process

Open your site URL on a web browser connected to the Internet and access your website. In that you get below web page.

Welcome to Doctro

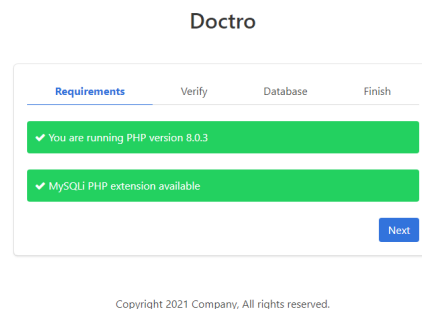
Thank you for purchasing Doctro
get your installation key and install database

Installer

Now start The Installation Process

When you click on Doctro installer you follow some easy step for Install database.

Step 1:



Check your server has valid php version and required extentions.

Step 2:

Once our system finds out that the server requirement is met properly it will proceed to next step which is license validation. there are lots of people selling infected code and to stop that we have implimented license security system. you will have to download license code from your envato account and your envato user name in order to validate the purchase

Requirement :

- Purchase Code
- Envato Username

Doctro

Requirements

Verify

Database

Finish

License code

enter your purchase/license code

Your name

enter your name/ervato username

Verify

Copyright 2021 Company. All rights reserved.

Step 3:

Once you have successfully validated your purchase now you have to enter those below details.

- Database Host
- Database username
- Database password
- Database name

Doctro

Requirements

Verify

Database

Finish

Database Host

enter your database host

Database Username

enter your database username

Database Password

enter your database password

Database Name

enter your database name

Import

Copyright 2021 Company. All rights reserved.

Step 4:

If you are at step 4 means you almost won the battle, , you just have to show your login admin email id and password which will be later used for logging into the backend panel.

Doctro

✔ Requirements ✔ Verify ✔ Database **Finish**

Admin credential

Admin Email

Admin Password

[Proceed to Login](#)

Copyright 2021 Company. All rights reserved.

After complete this step your installation is complete successfully and get your login screen after clicking on proceed to the login button.



Admin Login

Email

Password

Login

[Forgot Password?](#)

[Doctor Login](#)

[Pharamacy Admin Login](#)

Install PHP Script On cPanel

Steps To Install PHP Script On cPanel.

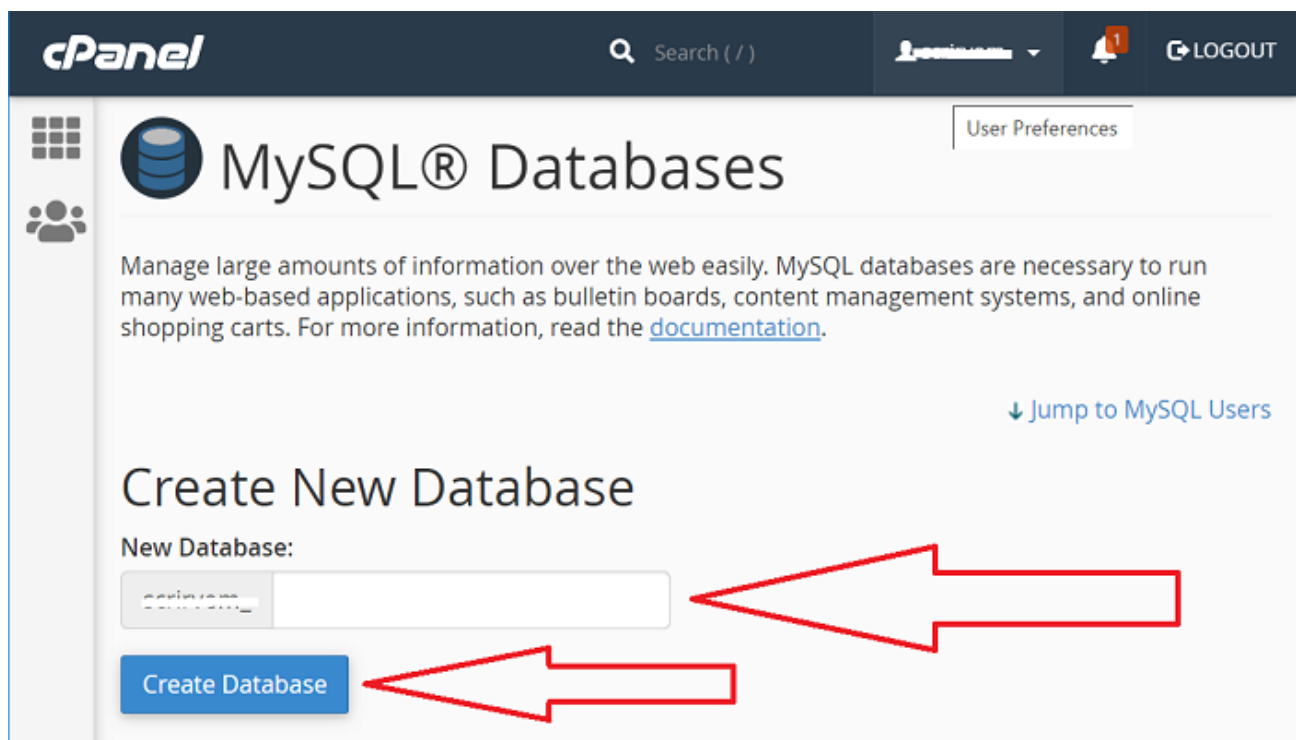
Step 1 :

Login into cPanel with the credentials. If you don't have a cPanel yet, simply purchase a Linux hosting account from any reliable hosting company of your choice. After your purchase, your cPanel login details will be sent to you.

For some hosting companies, you will have to log in from the client's side. That should not be a problem; either way, it's still taking linking to the same cPanel.

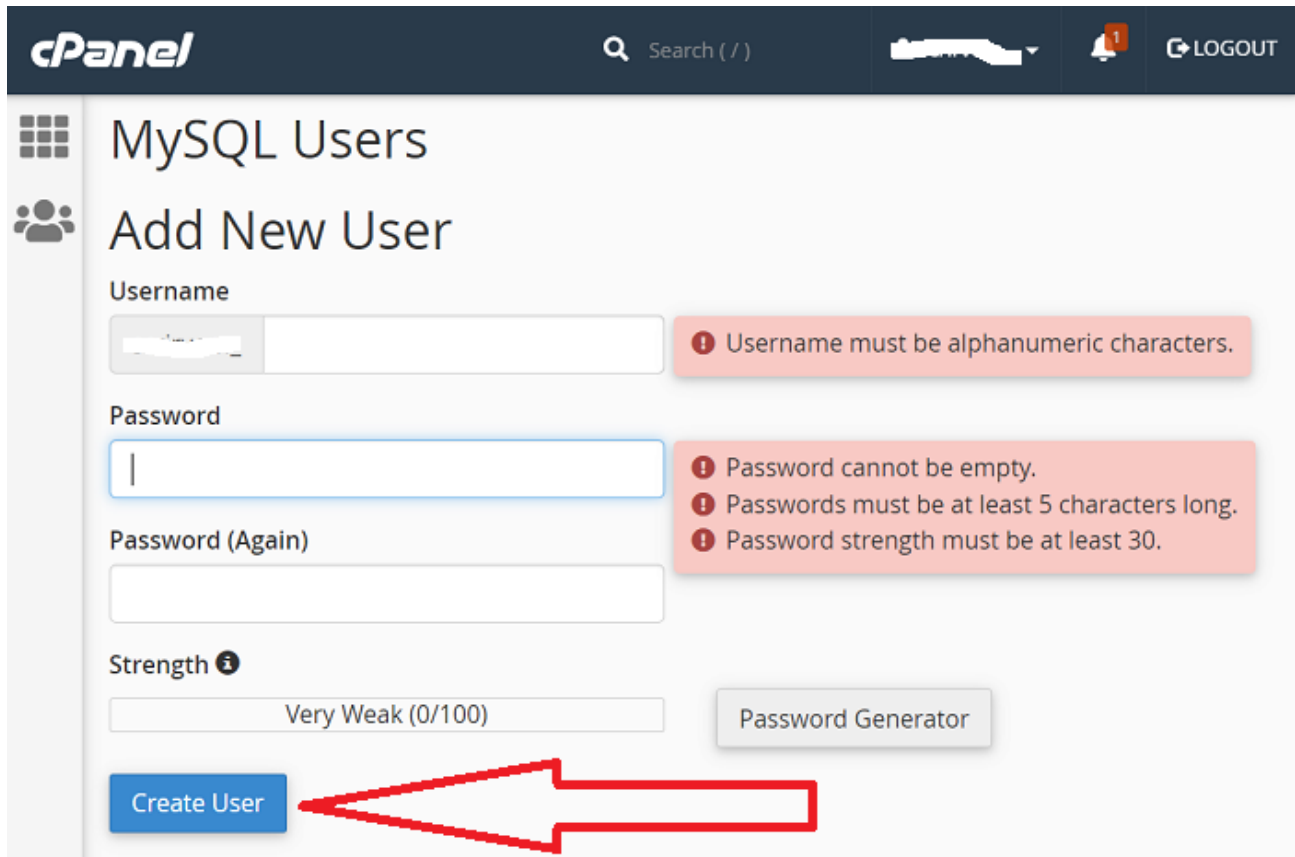
Step 2 :

After a successful login, **click on "MySQL Databases"** and create a database. Creating a database is as easy as giving it a name and click on **"Create Database."** The system will automatically create a database with the name you entered.



Step 3:

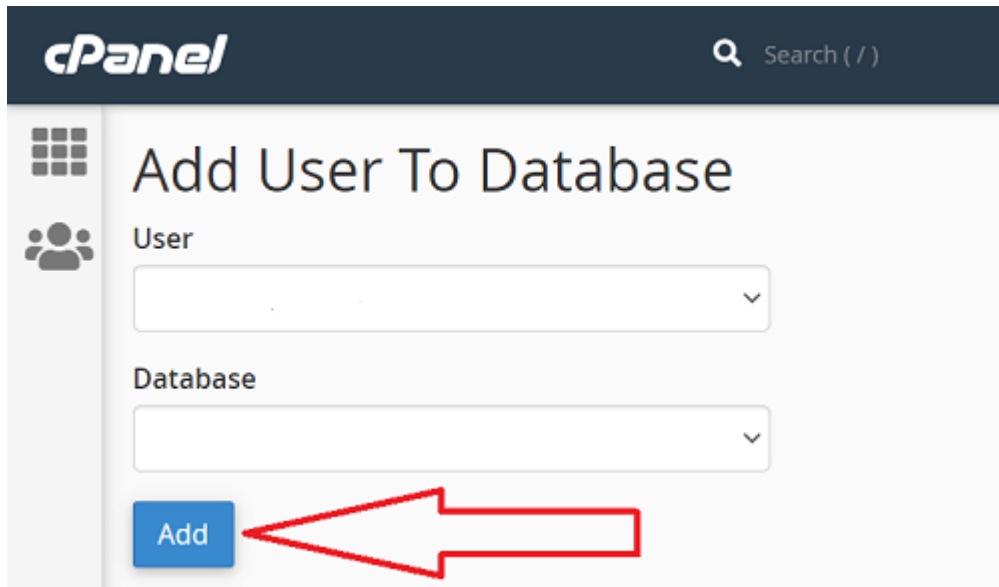
On the same page, you will have to **create a “MySQL Users.”** Scroll down the page, **create a user**, and **give it a password**.



The image shows the cPanel interface for adding a new MySQL user. The header includes the cPanel logo, a search bar, a language dropdown, a notification bell with one alert, and a LOGOUT button. The main content area is titled 'MySQL Users' and 'Add New User'. It contains three input fields: 'Username', 'Password', and 'Password (Again)'. The 'Username' field has a red error message: 'Username must be alphanumeric characters.' The 'Password' field has two red error messages: 'Password cannot be empty.' and 'Passwords must be at least 5 characters long.' The 'Password (Again)' field has a red error message: 'Password strength must be at least 30.' Below these fields is a 'Strength' indicator showing 'Very Weak (0/100)' and a 'Password Generator' button. At the bottom left is a blue 'Create User' button, which is highlighted by a large red arrow pointing towards it from the right.

Step 4:

For your script to be able to work with the database, you will have to **add the user you created to the database**, which was first created. To do this, scroll down on the same page you will see where it's written **“Add User to Database”** with two dropdowns. Simply select the database you created on the first dropdown, and select the created user on the second dropdown and click on **“Add.”**



Step 5:

After clicking on “**Add**,” it will take you to another where you will have to **check all the checkboxes and click on Make Changes**. This will give the user full access to the database via the script.

cPanel Search

Manage User Privileges

User:
Database:

☒ ALL PRIVILEGES

<input checked="" type="checkbox"/> ALTER	<input checked="" type="checkbox"/> ALTER ROUTINE
<input checked="" type="checkbox"/> CREATE	<input checked="" type="checkbox"/> CREATE ROUTINE
<input checked="" type="checkbox"/> CREATE TEMPORARY TABLES	<input checked="" type="checkbox"/> CREATE VIEW
<input checked="" type="checkbox"/> DELETE	<input checked="" type="checkbox"/> DROP
<input checked="" type="checkbox"/> EVENT	<input checked="" type="checkbox"/> EXECUTE
<input checked="" type="checkbox"/> INDEX	<input checked="" type="checkbox"/> INSERT
<input checked="" type="checkbox"/> LOCK TABLES	<input checked="" type="checkbox"/> REFERENCES
<input checked="" type="checkbox"/> SELECT	<input checked="" type="checkbox"/> SHOW VIEW
<input checked="" type="checkbox"/> TRIGGER	<input checked="" type="checkbox"/> UPDATE

Finally, your database is created. let's jump to the EventRight installation.

Mobile Application Setup

Flutter installation

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 2.2.2 stable release of the Flutter SDK:

```
$ cd ~/development$ unzip ~/Downloads/flutter_macos_vX.X.X-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 /Use
```

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$
```

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 bit code 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.

Create and run a simple Flutter app

To create your first Flutter app and test your setup, follow these steps:

1. Create a new Flutter app by running the following from the command line:
2. A `my_app` directory is created, containing Flutter's starter app. Enter this directory:
3. To launch the app in the Simulator, ensure that the Simulator is running and enter:

Deploy to iOS devices

To deploy your Flutter app to a physical iOS device you'll need to set up physical device deployment in XCode and an Apple Developer account. If your app is using Flutter plugins, you will also need the third-party Cocoa Pods dependency manager.

1. You can skip this step if your apps do not depend on [Flutter plugins](#) with native iOS code. [Install and set up Cocoa Pods](#) by running the following commands:

```
$ sudo gem install cocoapods
```

2. Follow the XCode signing flow to provision your project:
 1. Open the default XCode workspace in your project by running `open ios/Runner.xcworkspace` in a terminal window from your Flutter project directory.

2. Select the device you intend to deploy to in the device drop-down menu next to the run button.
3. Select the `Runner` project in the left navigation panel.
4. In the `Runner` target settings page, make sure your Development Team is selected under **Signing & Capabilities > Team**.

When you select a team, XCode creates and downloads a Development Certificate, registers your device with your account, and creates and downloads a provisioning profile (if needed).

- To start your first iOS development project, you might need to sign into Xcode with your Apple ID. Development and testing is supported for any Apple ID. Enrolling in the Apple Developer Program is required to distribute your app to the App Store. For details about membership types, see [Choosing a Membership](#).
- The first time you use an attached physical device for iOS development, you need to trust both your Mac and the Development Certificate on that device. Select `Trust` in the dialog prompt when first connecting the iOS device to your Mac.
Then, go to the Settings app on the iOS device, select **General > Device Management** and trust your Certificate. For first time users, you may need to select **General > Profiles > Device Management** instead.
- If automatic signing fails in XCode, verify that the project's **General > Identity > Bundle Identifier** value is unique.

3. Start your app by running `flutter run` or clicking the Run button in XCode [Android setup](#).

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 4.1 (API level 16) 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.

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 4.1 (API level 16) 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.

Setup An Editor With Flutter

Install the Flutter and Dart plugins

The installation instructions vary by platform.

Mac

Use the following instructions for macOS:

1. Start Android Studio.
2. Open plugin preferences (**Preferences > Plugins** as of v3.6.3.0 or later).
3. Select the Flutter plugin and click **Install**.
4. Click **Yes** when prompted to install the Dart plugin.
5. Click **Restart** when prompted.

Linux or Windows

Use the following instructions for Linux or Windows:

1. Open plugin preferences (**File > Settings > Plugins**).
2. Select **Marketplace**, select the Flutter plugin and click **Install**.

Install VS Code

VS Code is a lightweight editor with Flutter app execution and debug support.

- [VS Code](#), latest stable version

Install the Flutter and Dart plugins

1. Start VS Code.
 2. Invoke **View > Command Palette...**
 3. Type “install”, and select **Extensions: Install Extensions**.
 4. Type “flutter” in the extensions search field, select **Flutter** in the list, and click **Install**.
This also installs the required Dart plugin.
-

Validate your setup with the Flutter Doctor

1. Invoke **View > Command Palette...**
2. Type “doctor”, and select the **Flutter: Run Flutter Doctor**.
3. Review the output in the **OUTPUT** pane for any issues. Make sure to select Flutter from the dropdown in the different Output Options.

How to Change BASE_URL

For User Project

1. Open project code with android studio
2. From left pane : Select Project ->
3. here you can see all project folder
4. Select lib folder.
5. Open Apiservice.dart file.
6. Select project -> lib -> apiservice-> Apiservice.dart File.
7. in this file you can see baseUrl: "Enter_Your_Base_Url/public/api/" : replace your Server URL with it.
8. After replace URL.
9. at the bottom of android studio : TODO, Dart Analysis, Terminal buttons available.
10. click on Terminal button.
11. execute command : - **flutter pub get && flutter pub run build_runner build --delete-conflicting-outputs** wait for completion of the process.
12. Now follow steps of How to Generate APK file.

NOTE : Base URL for app is : Enter_Your_BaseUrl/public/api/

Run The App

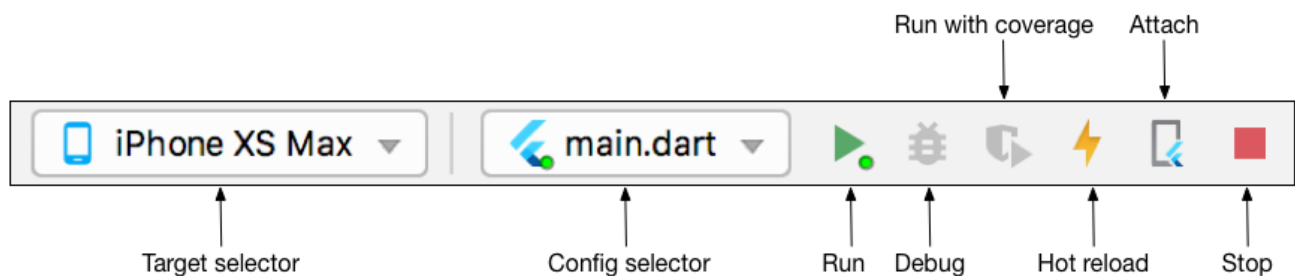
Using Android Studio

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

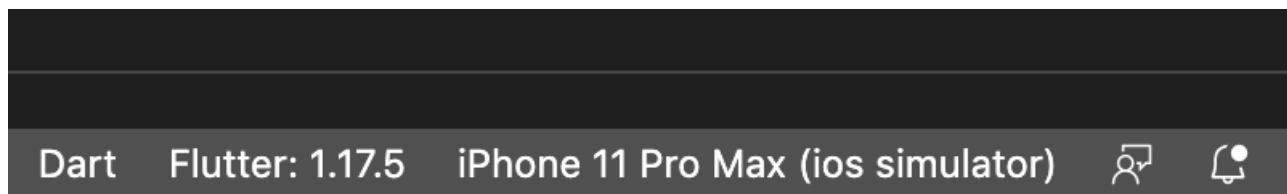
Using VS Code Studio

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.

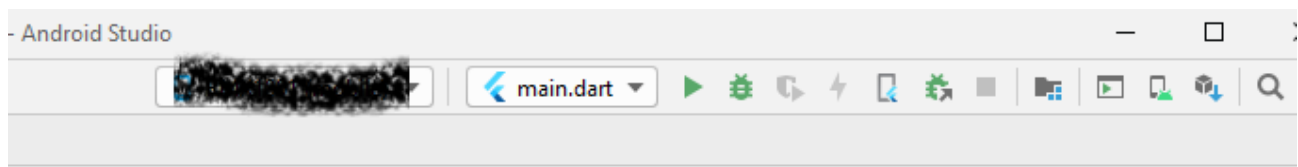
Starter app

Install App To Your Android Device

Install app to your android device

Steps for: How to install application in android device

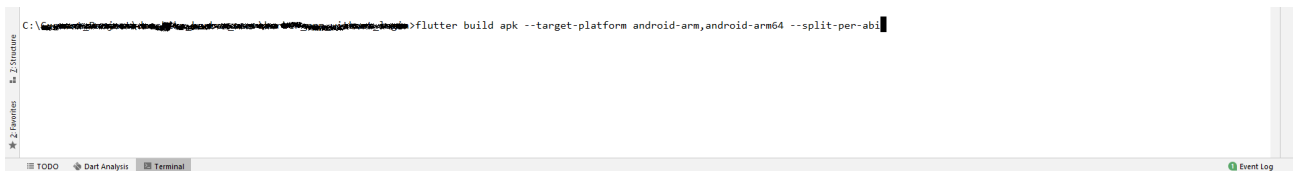
1. Connect your android device with OTG supported cable.
2. Open setting and turn on Developer option and turn on USB Debugging.
3. In android studio you can see your device connected at right hand side of screen.
4. then you need to just click on Green color play button to run it. it will take 2-4 minute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



How to Generate APK file

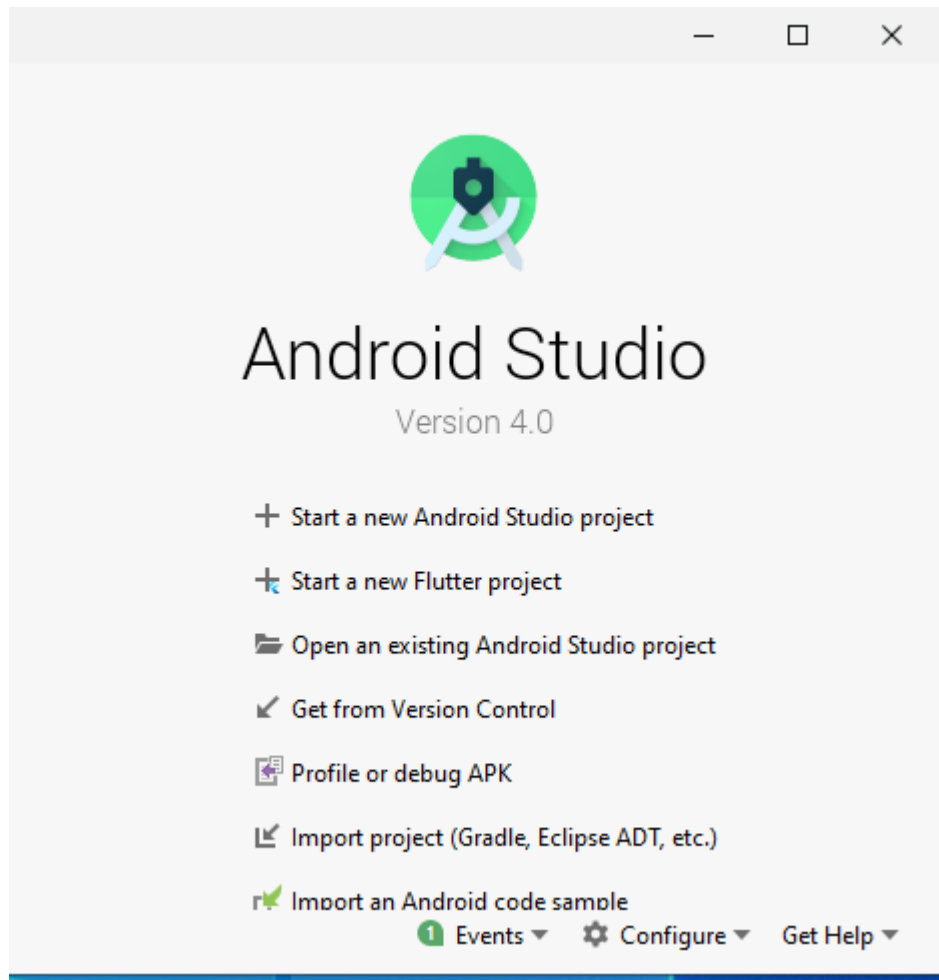
In the project folder there will have 2 folders located 1st for user and 2nd for driver, to generate both(user, driver) apk follow these steps.

1. at bottom of android studio : TODO, Dart Analysis, Terminal buttons available
2. click on Terminal button
3. execute command : - flutter build apk --target-platform android-arm,android-arm64 --split-per-abi
4. after 3 to 5 minute you can get APK file.



How to Generate Signed APK

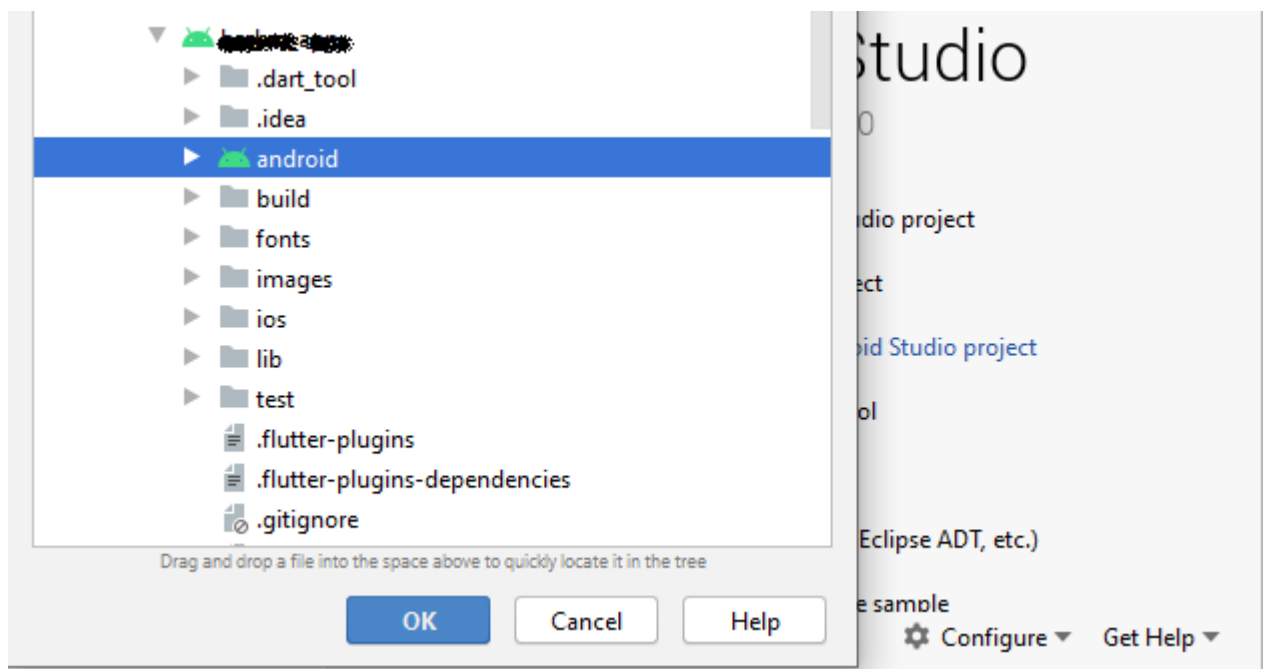
1. Open Android studio



2. Open an existing Android Studio project

3. Go to specific path of your code : MealUp -> android

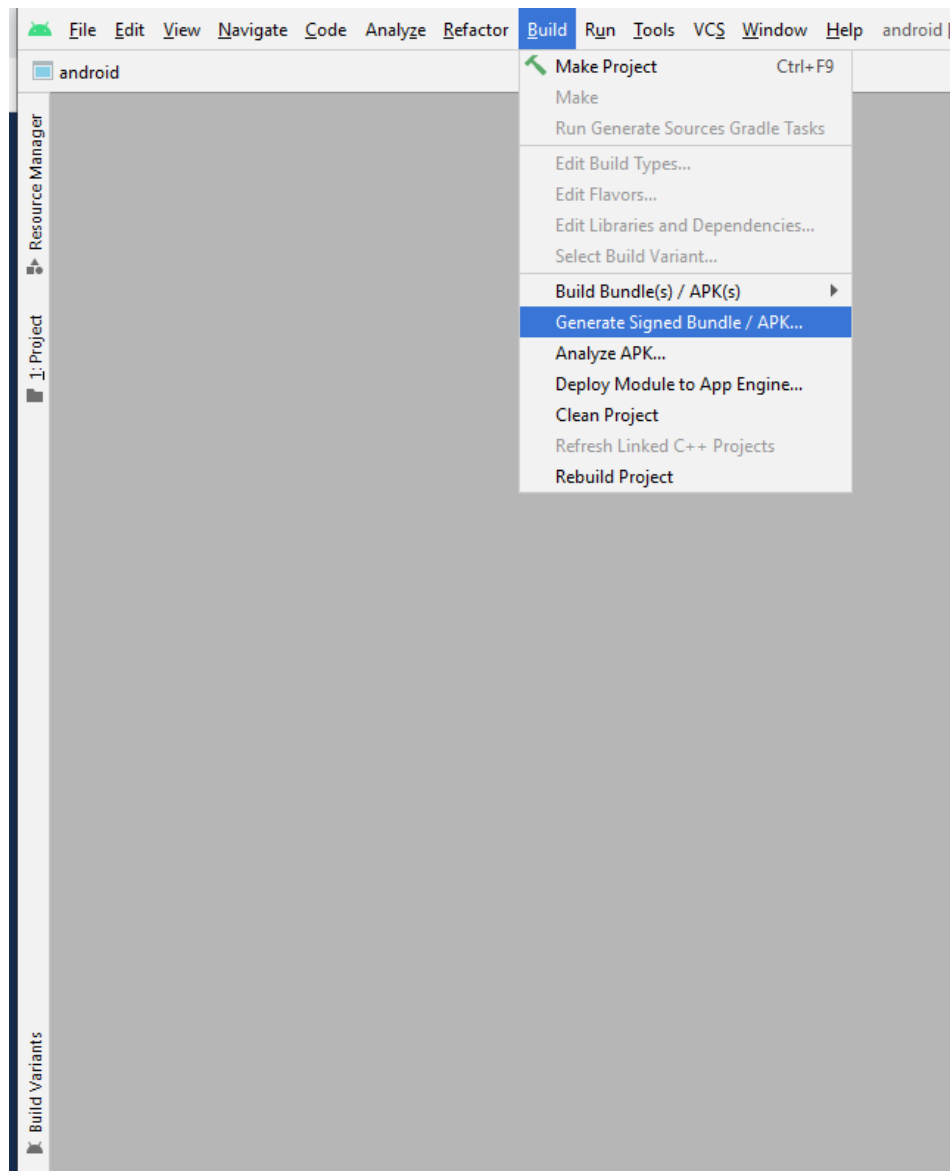
4. Select android



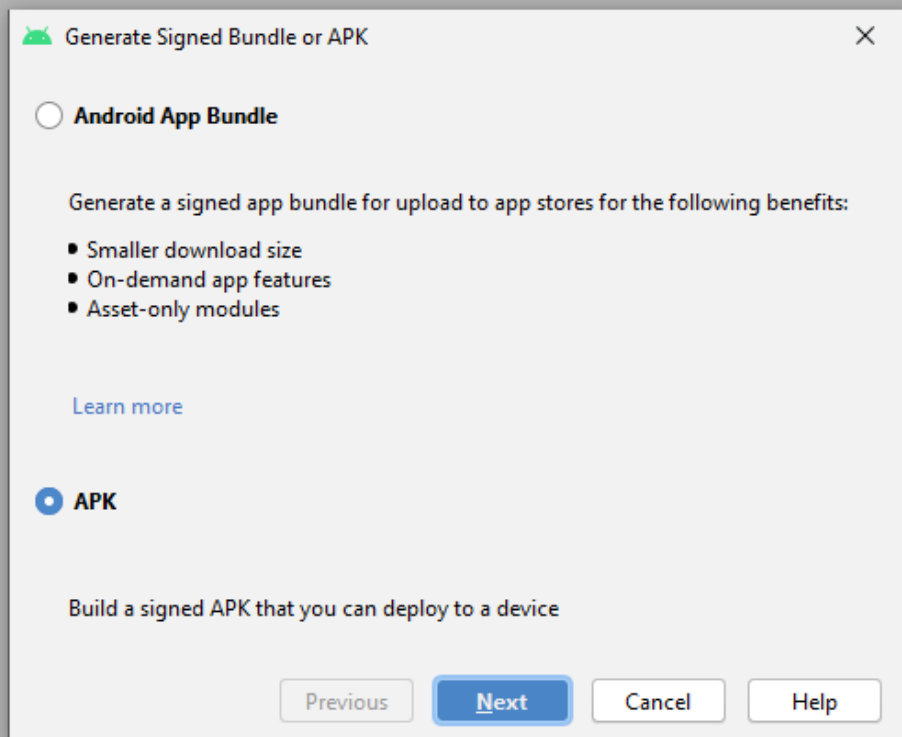
Click OK

Wait while build finish successfully.

After build finish :



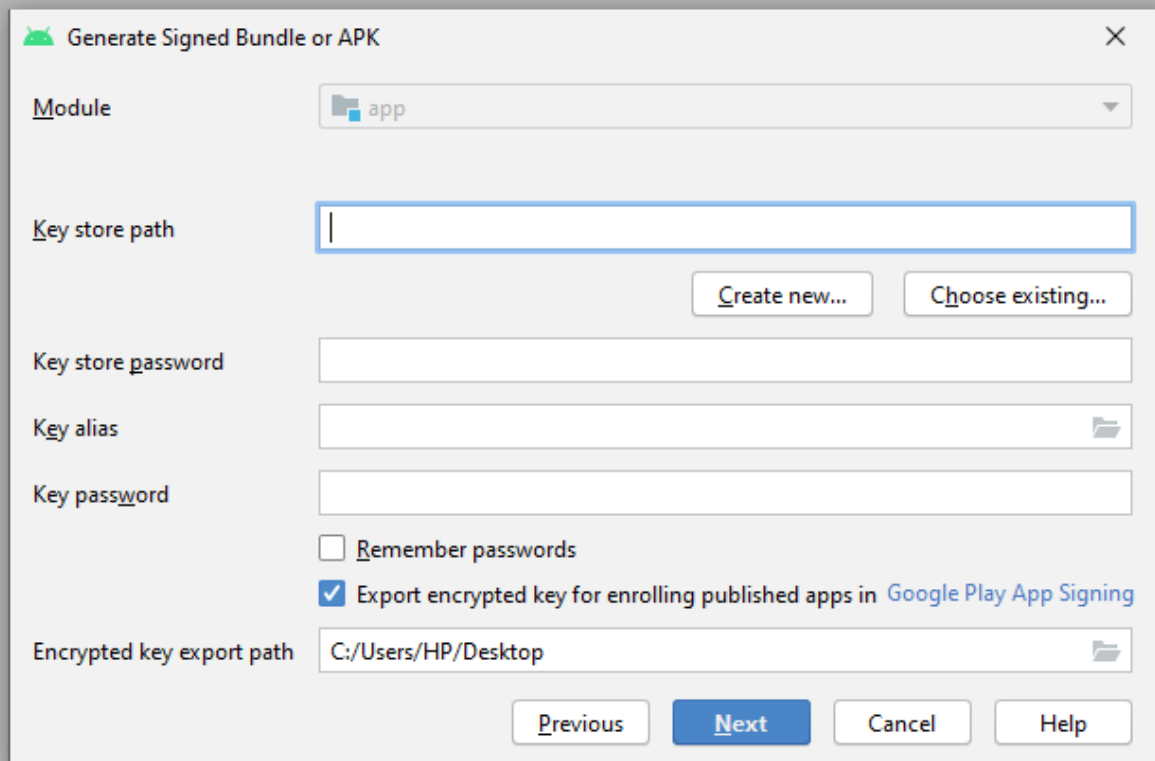
Select Build Menu -> Generate Signed Bundle/APK..



After Selecting Generate Signed Bundle/APK.. You can see above dialog box.

Here you have 2 option

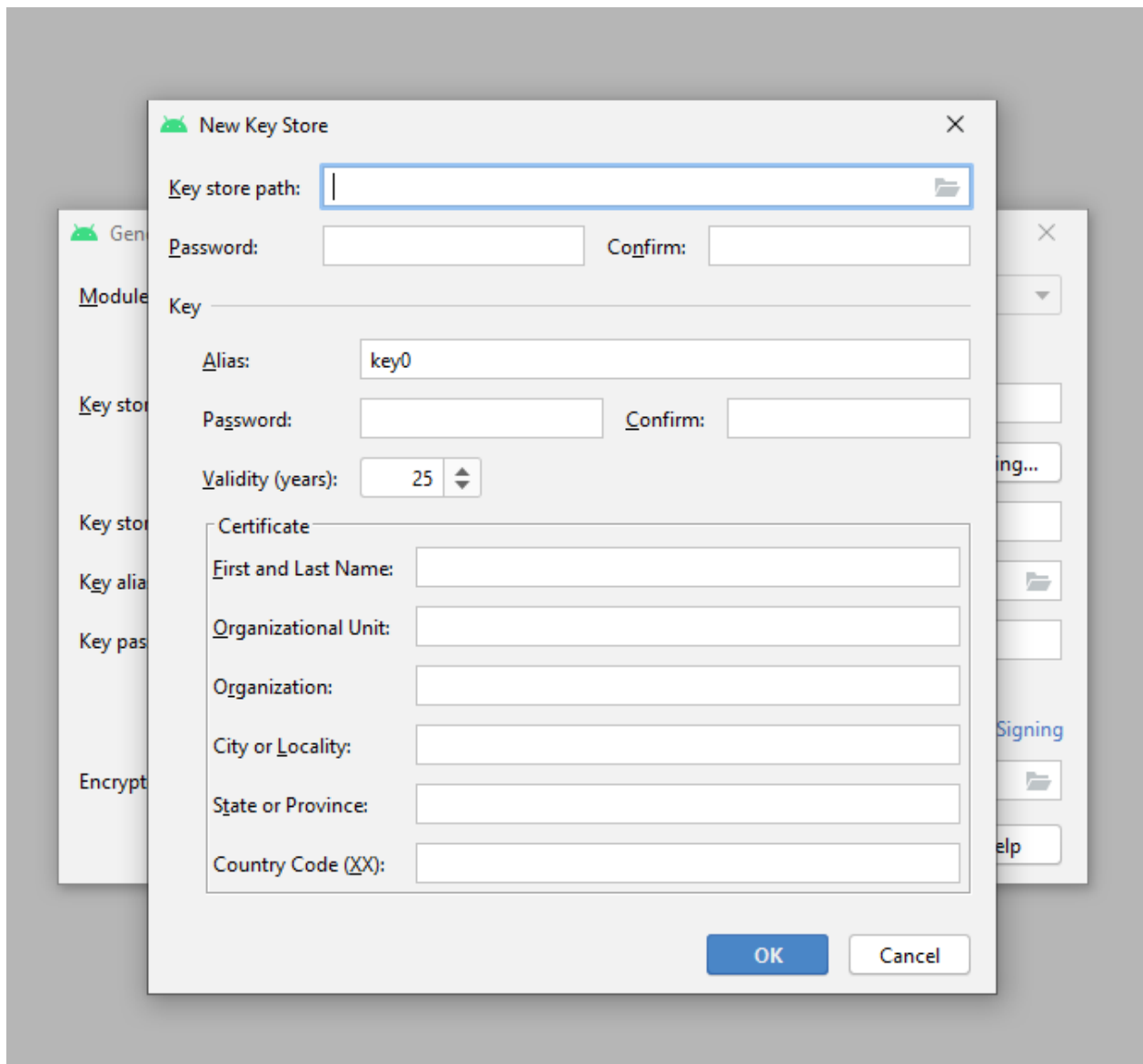
1. Android App Bundle
2. APK
3. Select App Bundle to Generate Signed APK.



You can see above dialog box :

If you are Generate first time signed apk at that time you need CREATE NEW KEYSTORE PATH

so that first click on Create new... After click create new... , You can see below dialog box



Fill Necessary Details : Key store path , Keystore Password Keystore 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 that keep it safe location and save it.)

Set Password and Confirm Password

Key——

Alias: Set Alias Name

Alias Password: Set Alias Password

Alias Confirm Password: Set Alias Confirm Password

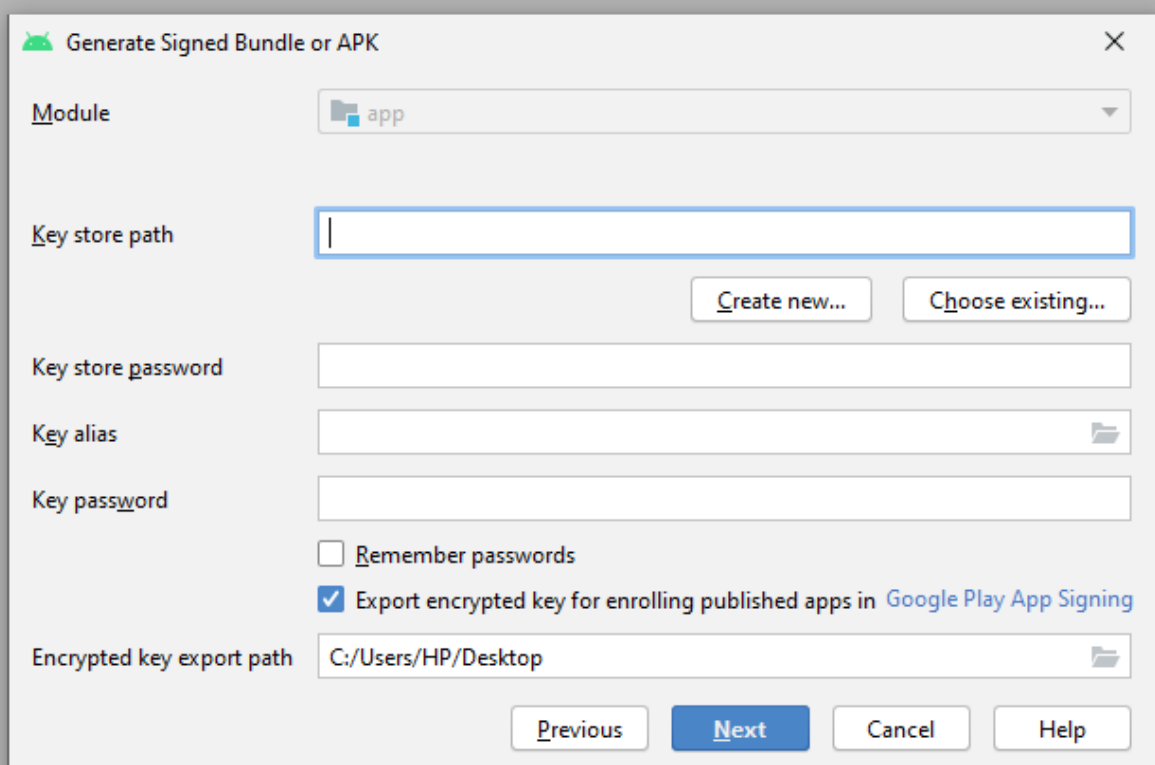
Validity : in years

Certificate :

First name and Last Name:

AFTER Fill UP All Details Click OK

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



Generate Signed Bundle or APK

Module: app

Key store path: [Empty field]

Key store password: [Empty field]

Key alias: [Empty field]

Key password: [Empty field]

☐ Remember passwords

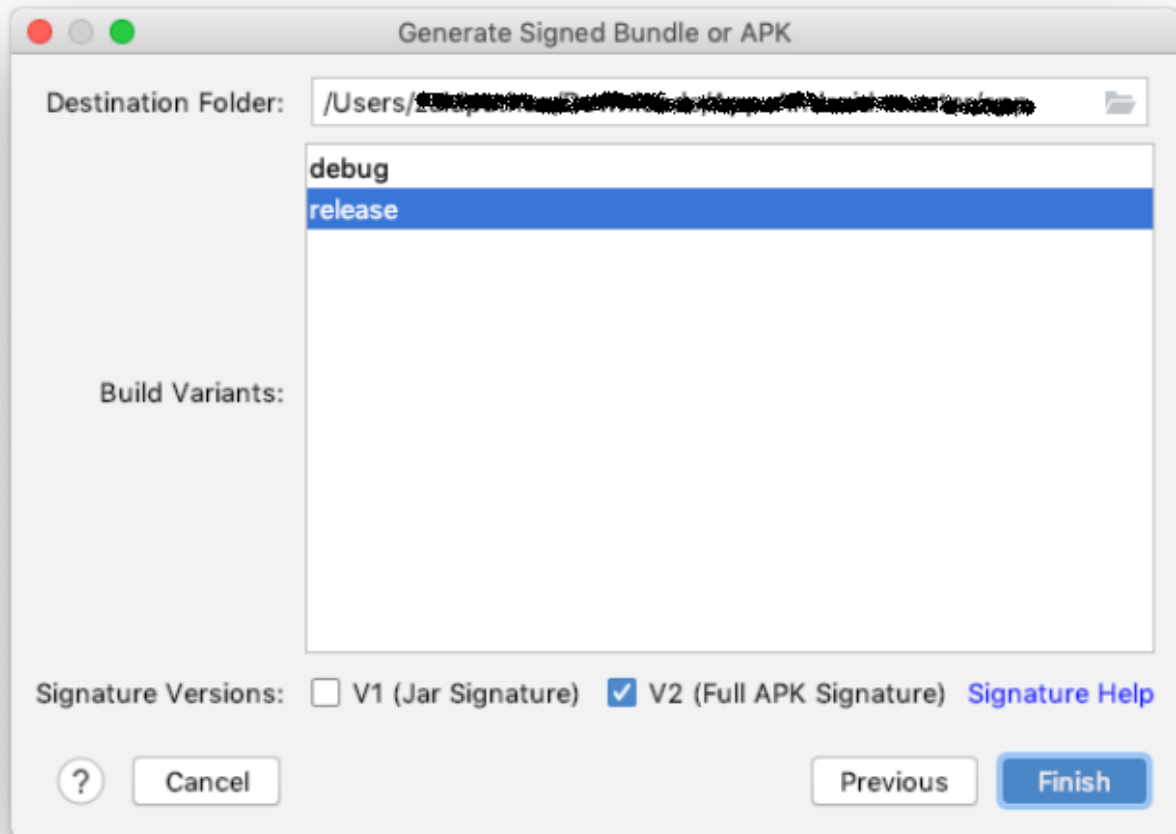
☒ Export encrypted key for enrolling published apps in [Google Play App Signing](#)

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

Buttons: Previous, Next, Cancel, Help

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

Click NEXT



Here you need to select build Variants : release

Signature versions : V1 and V2

You can select Signature version for secure apk file from reverse engineering.

Click Finish.

Generating Signed apk take 3-4 minute to Generate Signed APK.

How to publish APK to Google play store

How to publish APK to Google play store

Here steps for Android application published on the Google play store:

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

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 go throw the below link

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

[PreviousHow to publish APK to Google play store](#)

**shared on
codelist.cc**