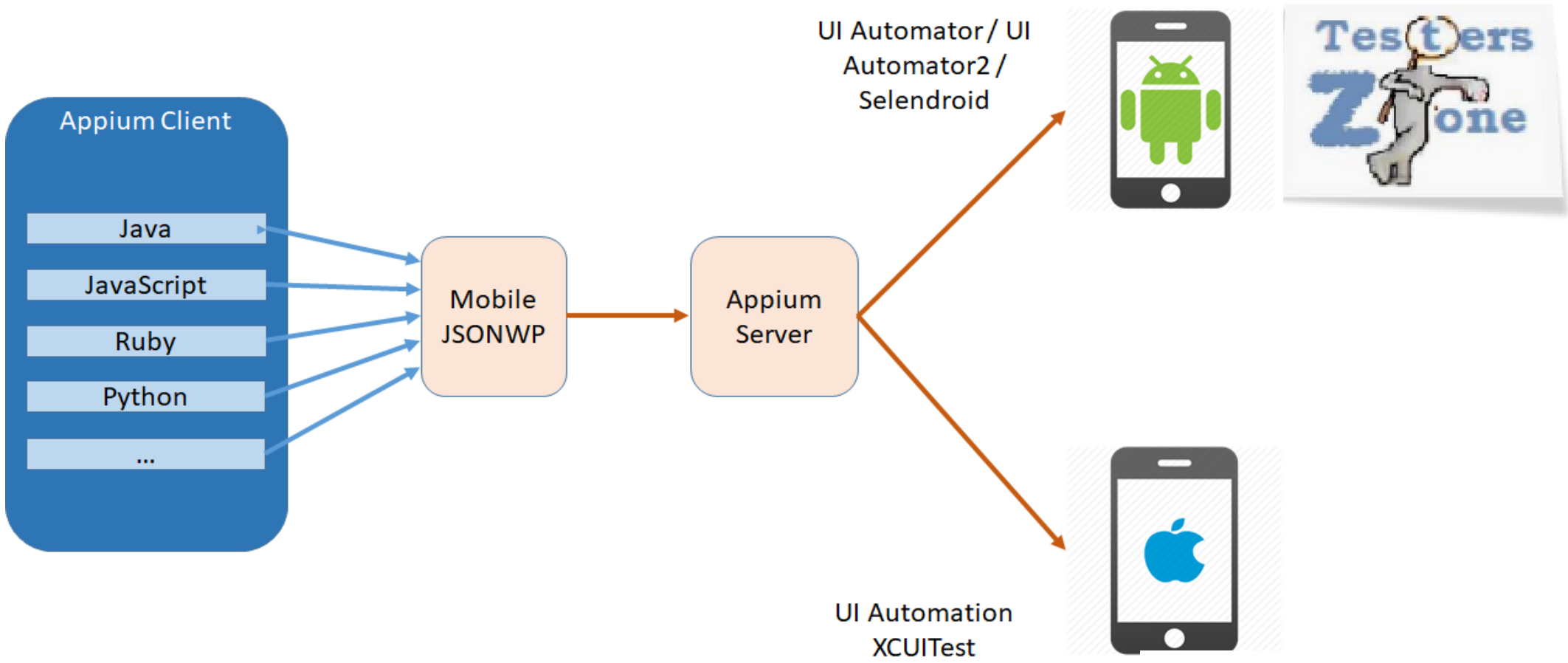


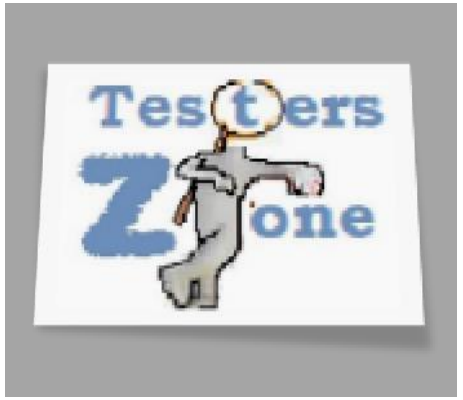
Appium set up on mac for iOS and Android

By Mithilesh Singh

List of things required :

- Home Brew
- Appium
- XCode
- XCode command line tool
- Install Carthage
- Appium Doctor

Install Home brew



- **It is a Package manager for macOS and is used to install software packages.**
- **Official URL:** https://brew.sh/
- I**nstall home brew using command line**:
  /bin/bash -c "$(curl -fsSL
  https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

  Note: better to verify the above command accessing home brew official URL or Copy directly from there.

Install Appium

- Appium can be installed using NPM and Appium Desktop.
- **via npm :**

  npm stands for node package manager, which helps to install node module using command line tool. Since appium is a node module we can install it using npm. First we need to install node using package manager brew, npm comes with node so no need to install it separately.
  Commands:
  brew install node
  npm install –g appium
- **Note: we can start appium server using command --> appium in the terminal.**
- **via Appium desktop:**

  it's a GUI representation. Can easily be installed from Appium official site.
  https://www.appium.io

Note:

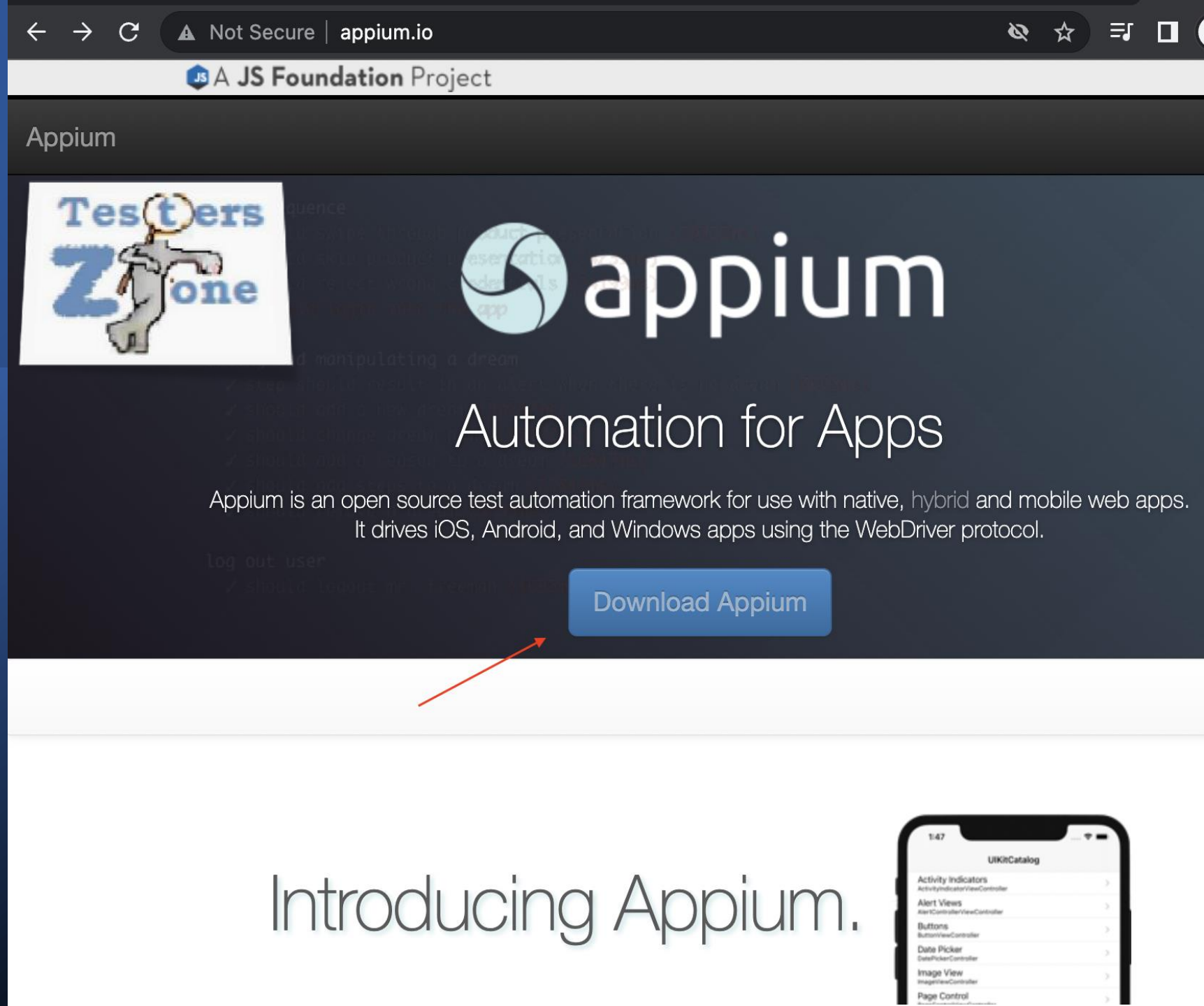
If we install appium using npm, we will get command line version of the appium, it will not have graphical interface. On the other hand if we install using appium desktop client we will have graphical user interface called appium inspector. It help to inspect the element of native mobile application.

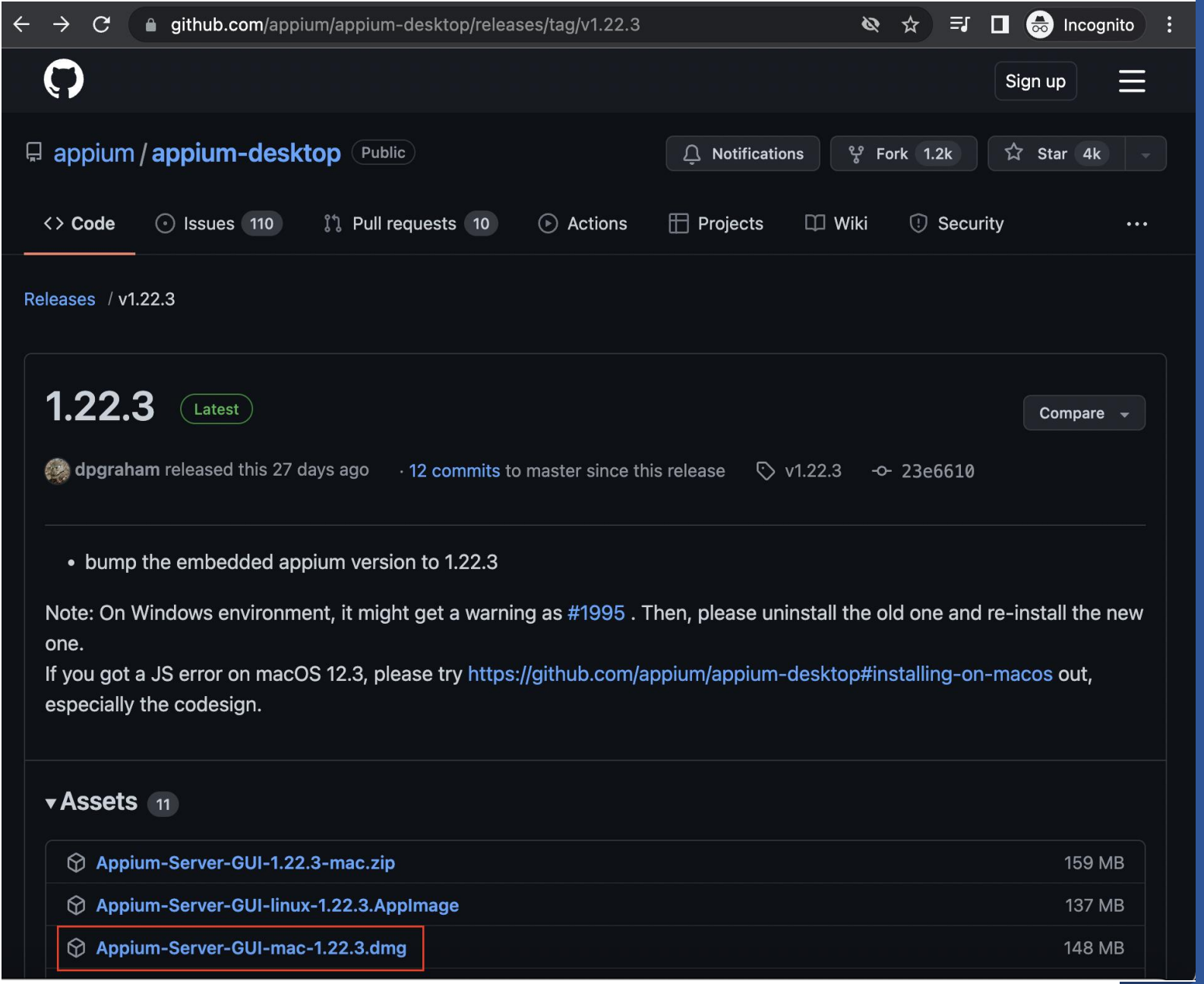
While installing appium using appium desktop client we no need to install node separately it comes as a bundle with appium inspector.

Steps to install using Appium Desktop client

Step 1:

Click on Download Appium





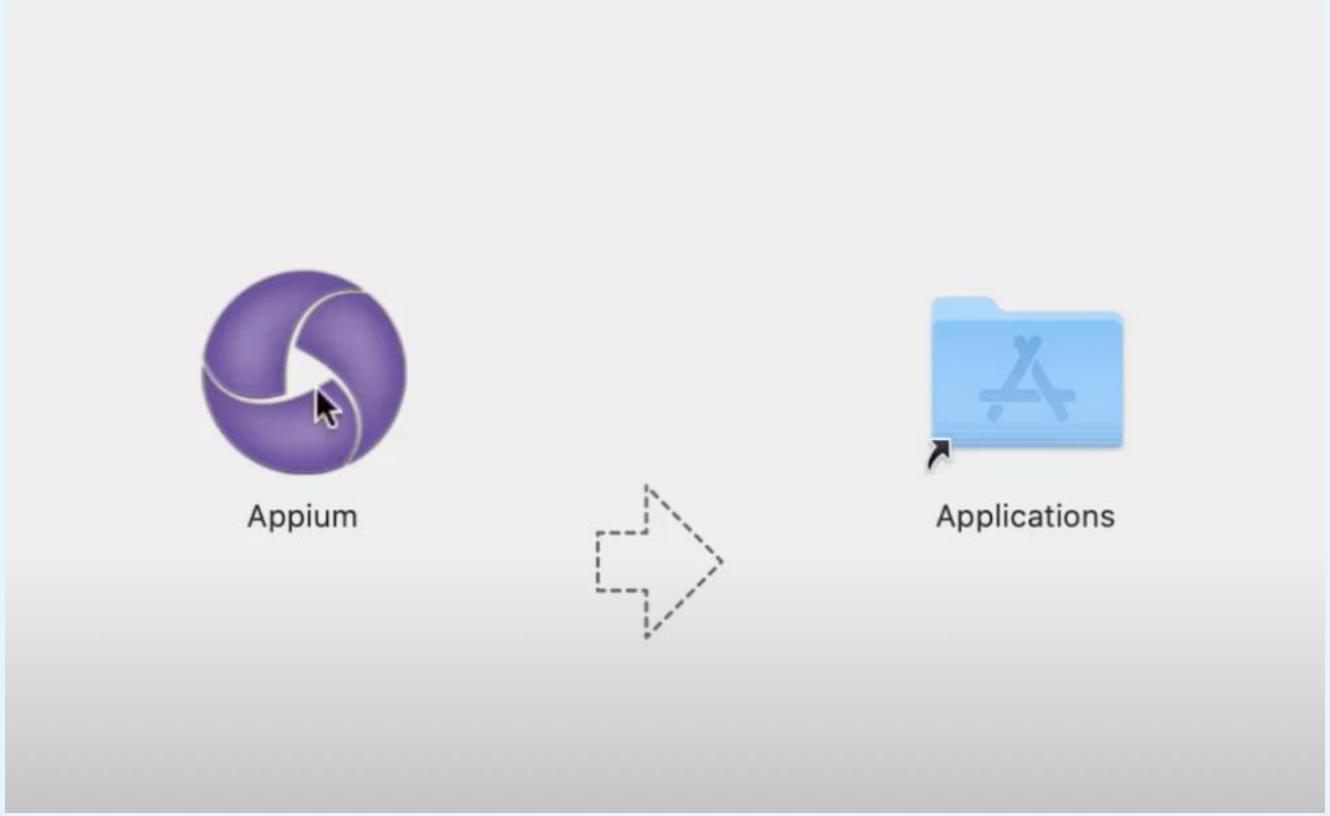
Step 2:

Click on the .dmg file, it will download the Appium on mac.

Note:

If user wants to download the older version then click on Release option present top left most corner below <> Code and select the older version.

Step: 3

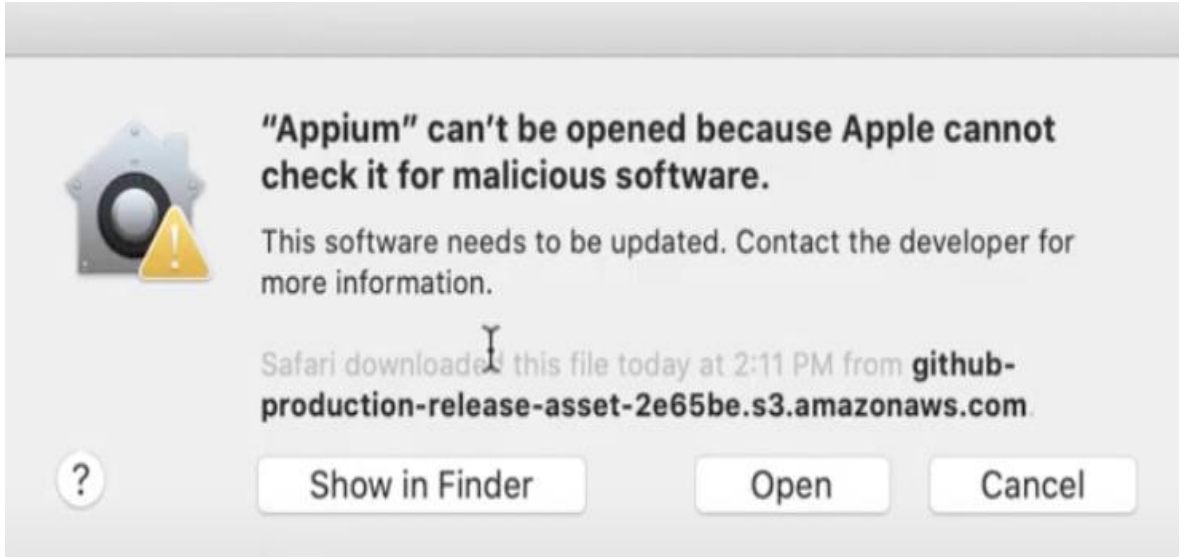


- Double click on downloaded .dmg file and once installation is complete move Appium server to Application directory by dragging.

Note:

- If you are launching Appium and getting below pop up, then

right click on Appium and click on open, same pop up will appear with open button. Click on open on the pop up



Install XCode

- It is a IDE(Integrated development environment for macOS for developing software for various platform like macOS, iOS, etc.
- We can search XCode in app store and install it on latest mac OS.
- To install the app from app store we need Apple ID.
- If you don't have apple id, while installing XCode you will be asked either enter apple id or create apple id. That time click on create apple id and fill the details as per screen.

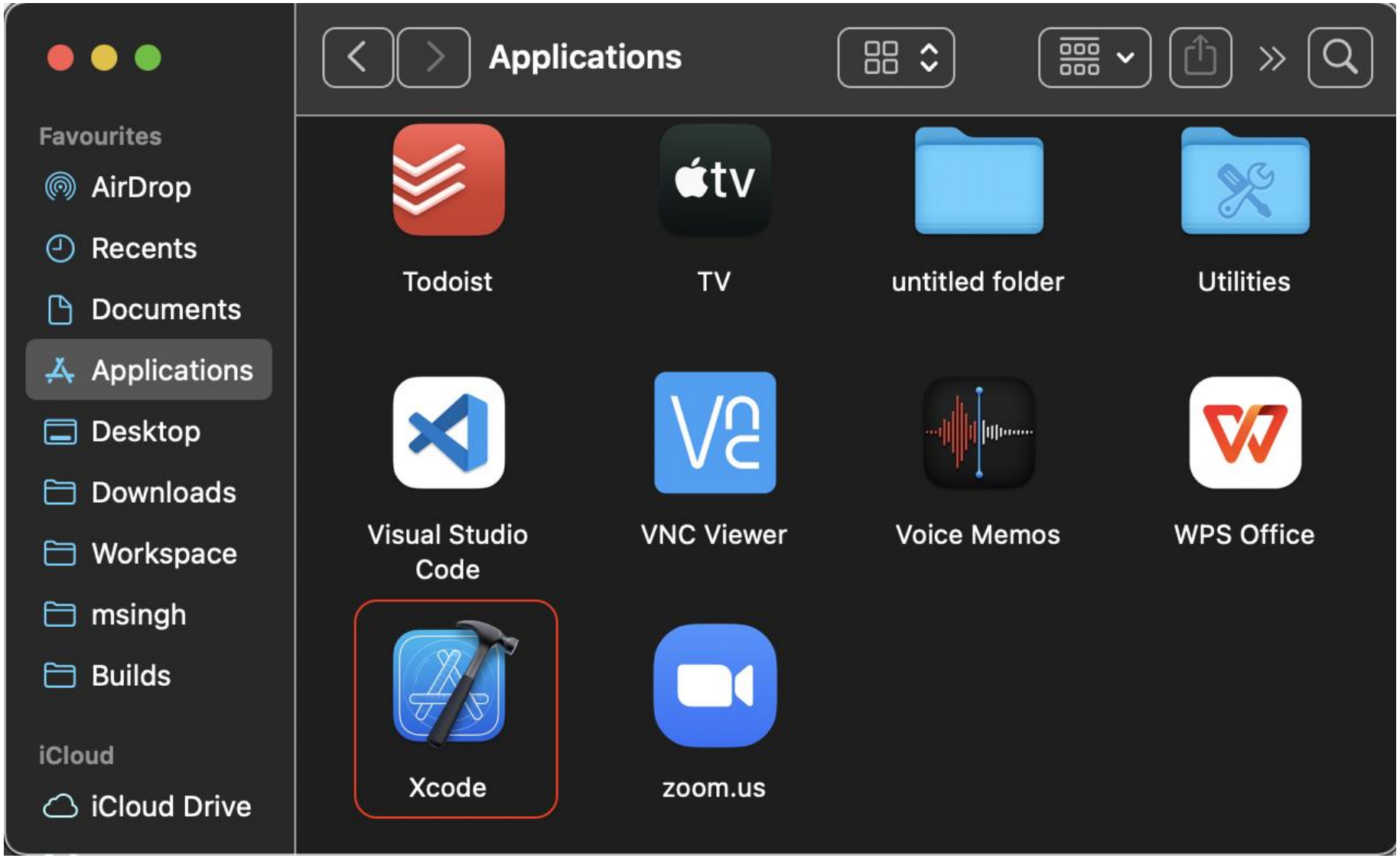
Screen: 1

If XCode is not installed then after searching it in app store it will open the XCode with get label. Once we click on get it converts into install label(observe in next screen).

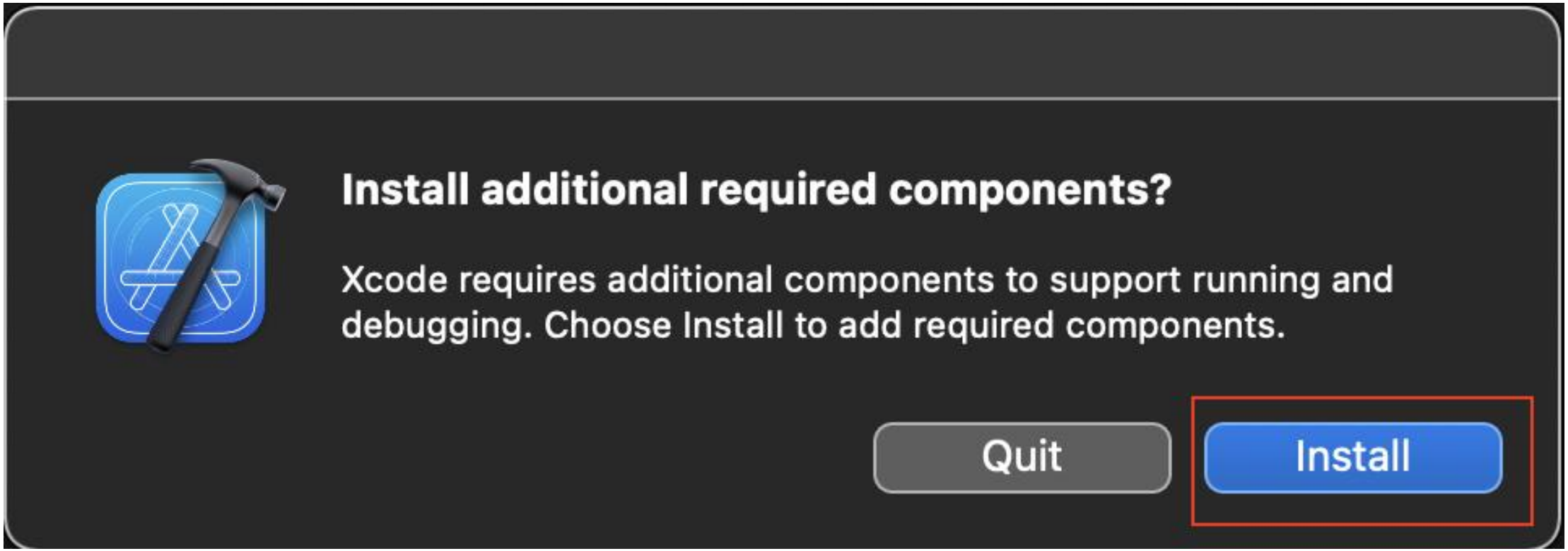
Screen: 2


Xcode
Developer Tools
INSTALL
YOUR SMOOTHIE IS READY!
Berry Blue is ready to be picked up.

Once installation is done you can open Application directory and search XCode



Once we click on XCode for very first time we need to install additional required components.



Allow XCode with your password to install additional components

XCode home page looks like this.
Note: You will get only webDriverAgent, other one(TestersZoneApp) is **created** by me.

Install XCode command line tool:

Install using command : xcode–select –install

If it is installed already, will get below screen..

```
xcode-select: error: command line tools are already installed, use "Software Upd
ate" to install updates
```

Install Carthage

Carthage is a *simple* dependency manager for macOS and iOS, created by a group of developers from GitHub

Not only was Carthage the first dependency manager to work with Swift, but it's also *written* in Swift! It exclusively uses dynamic frameworks, rather than static libraries.

We need this for WebDriverAgent.

Installation command: **brew install carthage**

Install APPIUM-DOCTOR

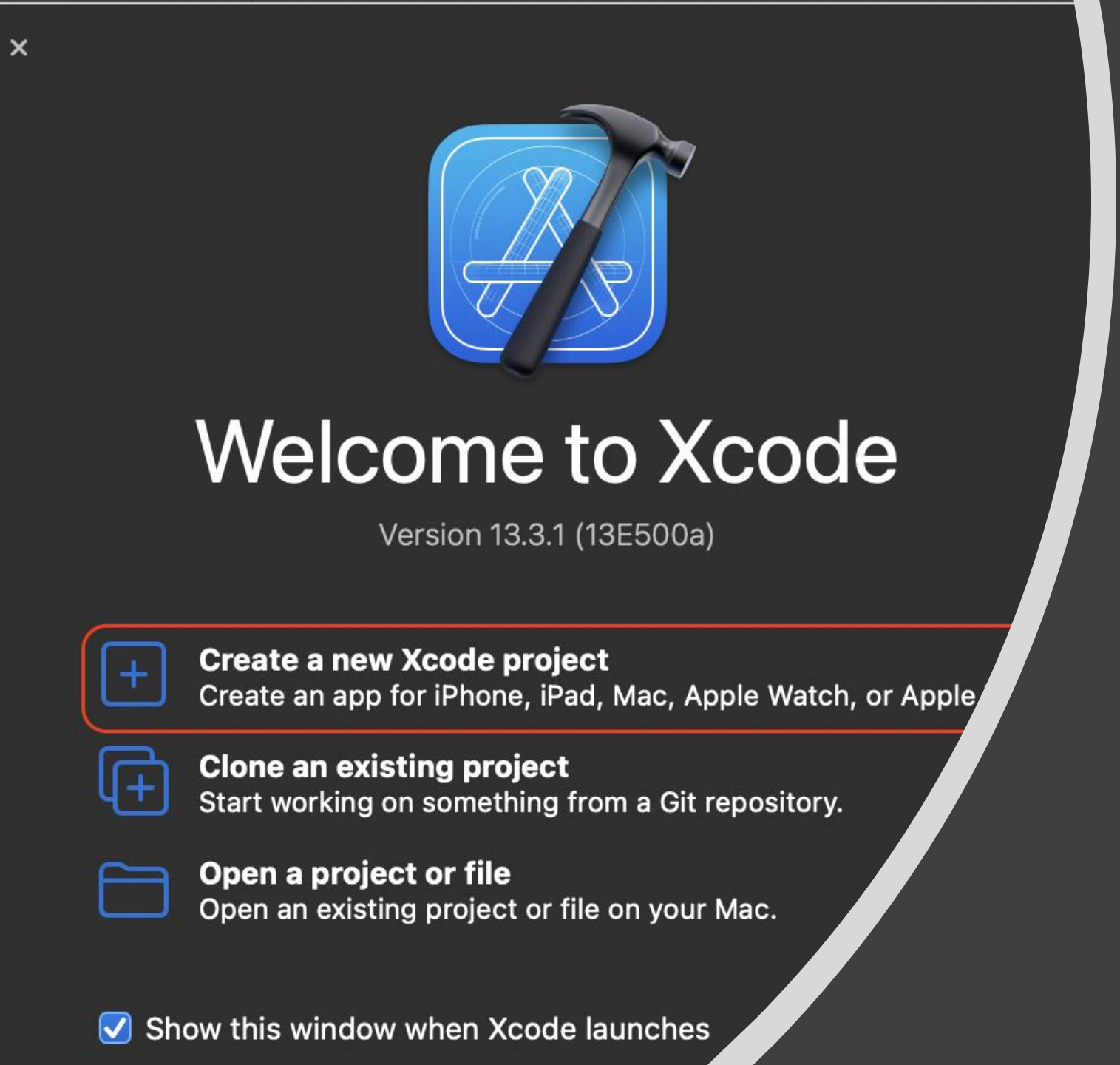
```
(base) ➜  ~ npm install -g appium-doctor
npm WARN deprecated authorize-ios@1.2.1: Moved into appium
npm WARN deprecated debug@4.1.1: Debug versions >=3.2.0 <3.2.7 || >=4 <4.
e a low-severity ReDos regression when used in a Node.js environment. It
mmended you upgrade to 3.2.7 or 4.3.1. (https://github.com/visionmedia/de
ues/797)

added 7 packages, removed 25 packages, changed 235 packages, and audited
kages in 26s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 8.1.3 -> 8.9.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.9.0
npm notice Run npm install -g npm@8.9.0 to update!
npm notice
(base) ➜  ~
```

- Appium doctor is one of the node module that helps us to diagnose and fix common Node, iOS and Android configuration issues before starting Appium.
- Installation Command: **npm install –g appium-doctor**

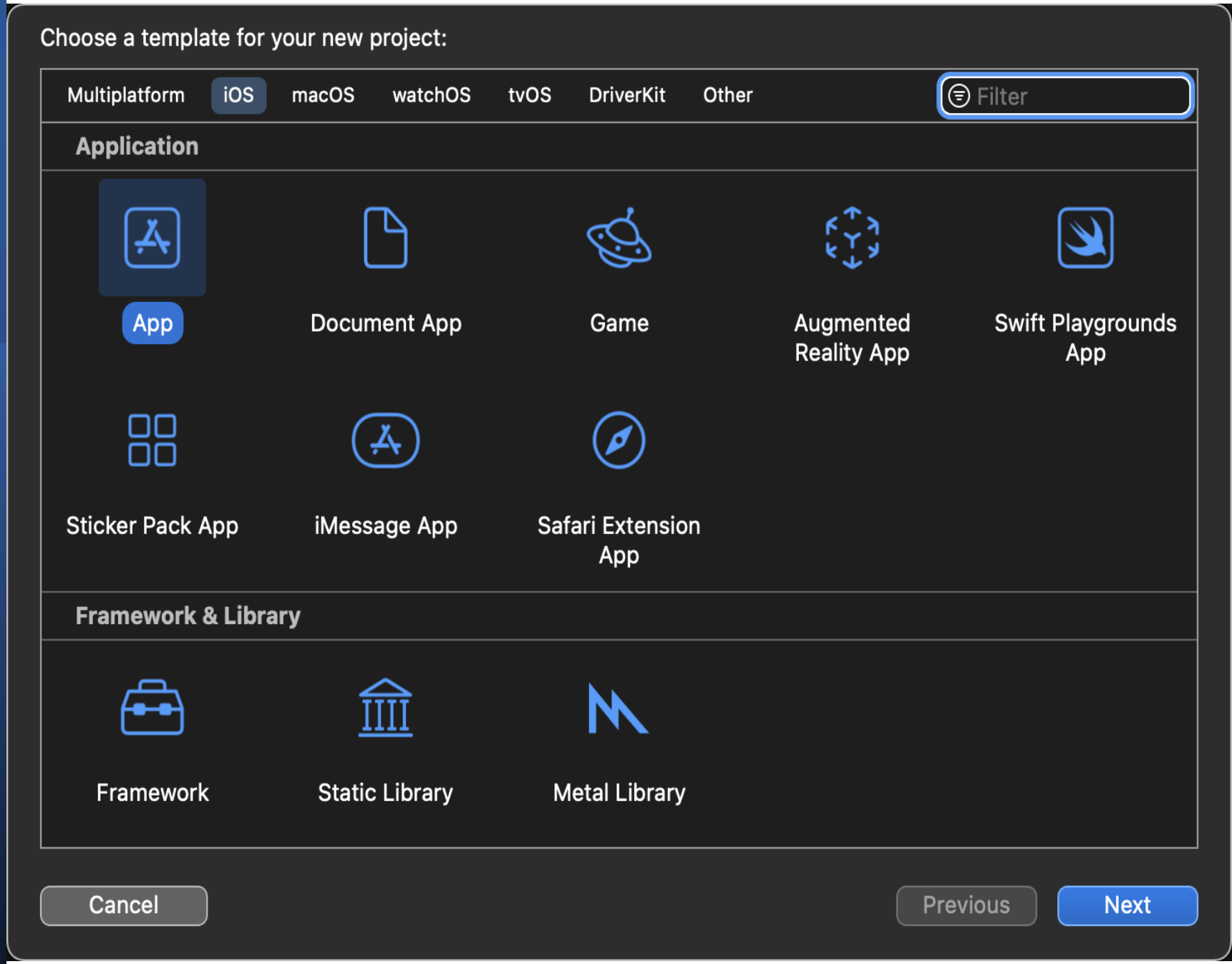


Create a dummy app and launching it in simulator.

Steps:1
Open the XCode and Click on Create a new XCode project

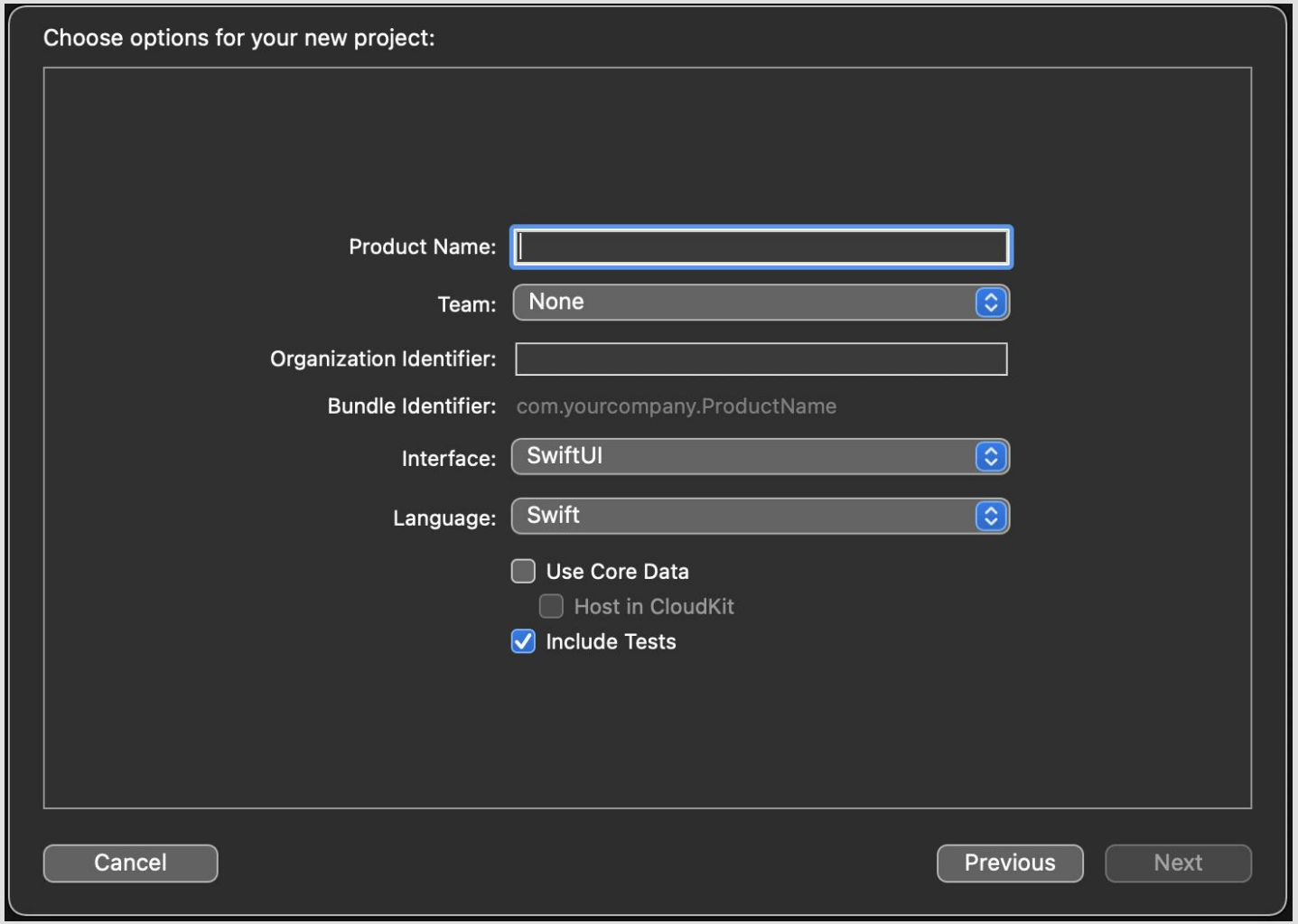
Step:2

Select iOS header and click on App as shown in screenshot



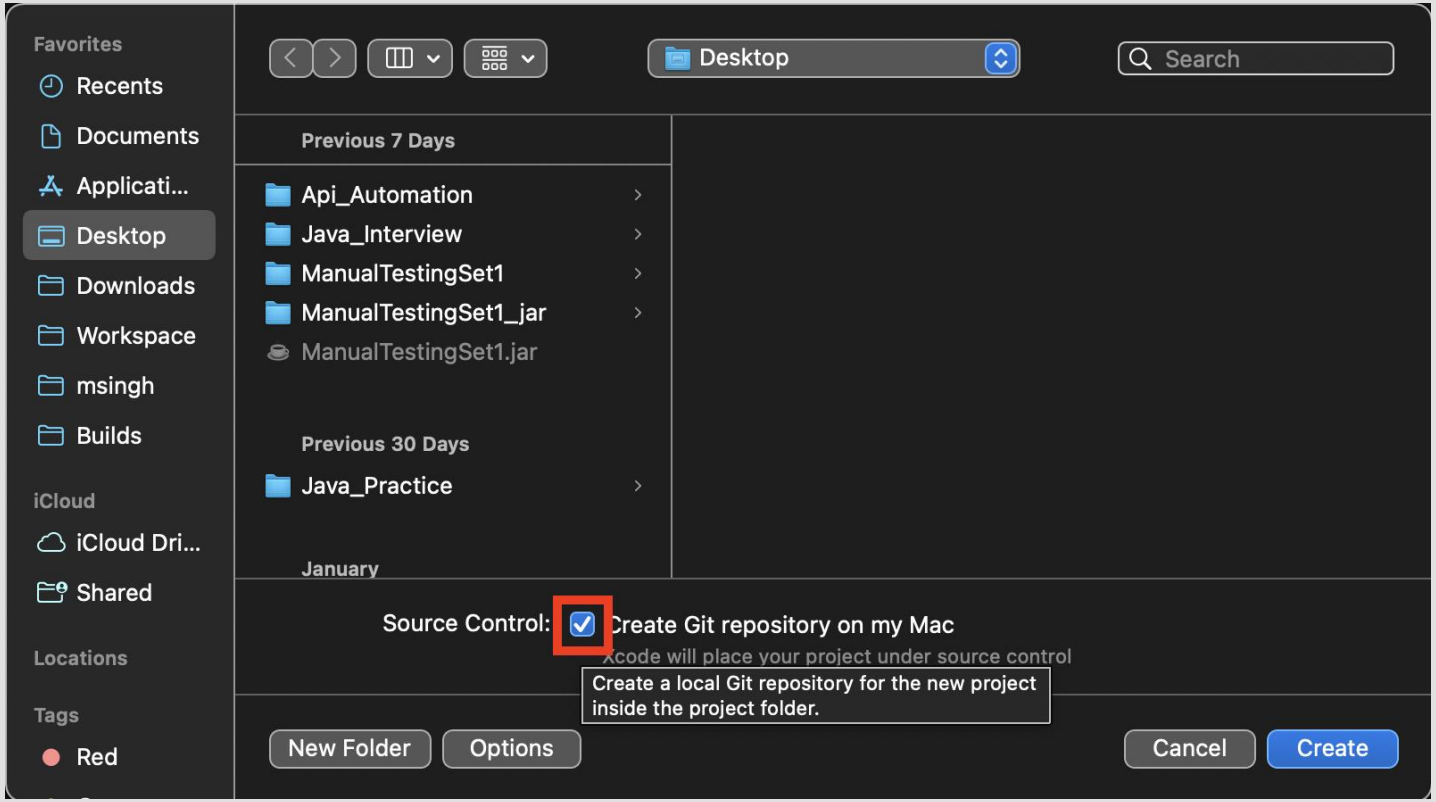
Step 3:

provide details and click Next button.



Step 4:

check the box and click on create



Your project will look like this ----->

TestersZoneA... | iPod to...eration) | TestersZoneApp: Ready

ContentView

TestersZoneApp > TestersZoneApp > ContentView > No Selection

- TestersZoneApp
  - TestersZoneApp
    - TestersZoneAppApp
    - ContentView
    - Assets
    - Preview Content
  - TestersZoneAppTests
    - TestersZoneAppTests
  - TestersZoneAppUITests
    - TestersZoneAppUITests
    - TestersZoneA...sLaunchTests

```
2  //  ContentView.swift
3  //  TestersZoneApp
4  //
5  //  Created by msingh on 07/05/22.
6  //

7
8  import SwiftUI
9
10 struct ContentView: View {
11     var body: some View {
12         Text("Hello, world!")
13             .padding()
14     }
15 }
16
17 struct ContentView_Previews: PreviewProvider {
18     static var previews: some View {
19         ContentView()
20     }
21 }
22
```

Filter

Line: 22 Col: 1

Identity and Type

Name: ContentView.swift

Type: Default - Swift Source

Location: Relative to Group

ContentView.swift

Full Path: /Users/msingh/Desktop/TestersZoneApp/TestersZoneApp/ContentView.swift

On Demand Resource Tags

Only resources are taggable

Target Membership

- [x] TestersZoneApp
- [ ] TestersZoneAppTests
- [ ] TestersZoneAppUITests

Text Settings

Text Encoding: No Explicit Encoding

Line Endings: No Explicit Line Endings

Indent Using: Spaces

Widths: Tab 4, Indent 4

- [x] Wrap lines

Build the app for the simulator.

Steps:

Open terminal

1. Use command -->

xcodebuild –showsdks

2. copy the name of iphonesimulator sdk, in our case it is

–sdk iphonesimulator15.4

```
msingh@ces-MacBook-Pro:~
(base) ➜  ~ xcodebuild -showsdks
2022-05-07 00:59:14.776 xcodebuild[90423:3775819] Requested but did not find extension point with identifier Xcode
.IDEKit.ExtensionSentinelHostApplications for extension Xcode.DebuggerFoundation.AppExtensionHosts.watchOS of plug
-in com.apple.dt.IDEWatchSupportCore
2022-05-07 00:59:14.776 xcodebuild[90423:3775819] Requested but did not find extension point with identifier Xcode
.IDEKit.ExtensionPointIdentifierToBundleIdentifier for extension Xcode.DebuggerFoundation.AppExtensionToBundleIden
tifierMap.watchOS of plug-in com.apple.dt.IDEWatchSupportCore
DriverKit SDKs:
        DriverKit 21.4                  -sdk driverkit21.4

iOS SDKs:
        iOS 15.4                        -sdk iphoneos15.4

iOS Simulator SDKs:
        Simulator - iOS 15.4            -sdk iphonesimulator15.4

macOS SDKs:
        macOS 12.3                      -sdk macosx12.3

tvOS SDKs:
        tvOS 15.4                       -sdk appletvos15.4

tvOS Simulator SDKs:
        Simulator - tvOS 15.4           -sdk appletvsimulator15.4

watchOS SDKs:
        watchOS 8.5                     -sdk watchos8.5

watchOS Simulator SDKs:
        Simulator - watchOS 8.5         -sdk watchsimulator8.5
```

Step 2:

xcodebuild –sdk <simulator_Name>

Note: first navigate to the project directory using cd <path> then use above command.

We will get the app path also which we can use to install the app in other simulators.

```
RegisterExecutionPolicyException /Users/msingh/Desktop/TestersZoneApp/build/Release-iphonesimulator/TestersZoneApp.a
pp (in target 'TestersZoneApp' from project 'TestersZoneApp')
    cd /Users/msingh/Desktop/TestersZoneApp
    builtin-RegisterExecutionPolicyException /Users/msingh/Desktop/TestersZoneApp/build/Release-iphonesimulator/Test
ersZoneApp.app

Touch /Users/msingh/Desktop/TestersZoneApp/build/Release-iphonesimulator/TestersZoneApp.app (in target 'TestersZoneA
pp' from project 'TestersZoneApp')
    cd /Users/msingh/Desktop/TestersZoneApp
    /usr/bin/touch -c /Users/msingh/Desktop/TestersZoneApp/build/Release-iphonesimulator/TestersZoneApp.app

** BUILD SUCCEEDED **

(base) ➜  TestersZoneApp
```

How to launch the App in the simulator? Using appium inspector with desired capabilities?

Appium Server | *Select Cloud Providers*

Remote Host: 127.0.0.1 | Remote Port: 4723

Remote Path: /wd/hub | SSL

> Advanced Settings

Desired Capabilities | Saved Capability Sets 1 | Attach to Session...

deviceName	text	iPhone 11
platformName	text	iOS
platformVersion	text	14.3
app	text	/Users/msingh/Desk
noReset	boolean	true
automationName	text	XCUITest

Automatically add necessary Appium vendor prefixes on start

JSON Representation

```
{
  "deviceName": "iPhone 11",
  "platformName": "iOS",
  "platformVersion": "14.3",
  "app":
"/Users/msingh/Desktop/TestersZoneApp/build/Release-
iphonesimulator/TestersZoneApp.app",
  "noReset": true,
  "automationName": "XCUITest"
}
```

Desired Capabilities Documentation

Save As... | Start Session

Set up Appium for real iOS device

- let's connect the iPhone with mac book and get the UDID of the phone.
- To get the UDID we can open the terminal and use command: **instruments –s devices** but if you are using latest XCode version 13 or above, instruments util will not be available there we have to use command: **xcrun xctrace list devices**

Web driver Agent?

- WebDriverAgent is a WebDriver server implementation for iOS that can be used to remote control iOS devices. It allows you to launch & kill applications, tap & scroll views or confirm view presence on a screen. This makes it a perfect tool for application end-to-end testing or general purpose device automation. It works by linking XCTest.framework and calling Apple's API to execute commands directly on a device. WebDriverAgent is developed and used at Facebook for end-to-end testing and is successfully adopted by Appium.

Code Signing

- When downloading software from the Internet, consumers must always be wary of 3rd parties masquerading as the software provider. With a resource like code signing, software can be assured it is coming from the proper source. Code signing is an operation where a software developer or distributor digitally signs the file being sent out, to assure users that they are receiving software that does what the creator says it will. The signature acts as proof the code has not been tampered with or modified from its original form.

Code signing process:

- Before the developers can sign their work, they need to generate a public/private key pair. This is often done locally through software tools such as `openssl`. Developers then give the public key and the organization's identity information to a trustworthy CA. The CA verifies the authenticity of identity information and then issues the certificate to the developer. This is the code signing certificate which was signed by CA's private key and contains the developer organization's identity and the developer's public key.

- When developers are ready to "sign" their work to establish authorship, they take all the code they wrote and they hash it. The value that spits out is then encoded using the abovementioned private key (usually generated by the author), along with the code signing certificate that contains the public key and the identity of the author (proving the authorship). The output of this process is then added to the software to be shipped out.

- This constitutes a code signing operation. The public key of the CA is already pre-installed in most browsers and operating system trust stores. When a user downloads the software, they use the CA's public key to first verify the authenticity of the code signing certificate embedded in the signed software to confirm that it's from a trustworthy CA. The developer's public key is then extracted from the certificate and used to decrypt the encrypted hash.

- Then, the software is hashed again, and the new value is compared to the decrypted one. If the user's hash value and the developer's hash value match, then the software hasn't been corrupted or tampered with during transmission. The user is then alerted that the software is as the developer last left it, and (if the developer is to be trusted) it's safe to install and run.

Why Provisioning Profiles?

Unlike Android, you can't install any app on an iOS device. It has to be signed by Apple first. However, when you're *developing* an app, you probably want to test it before sending it to Apple for approval. Provisioning profile act as a link between the device and the developer account. During development, you choose which devices can run your app and which app services your app can access. A provisioning profile is downloaded from your developer account and embedded in the app bundle, and the entire bundle is code-signed. A Development Provisioning Profile must be installed on each device on which you wish to run your application code. If the information in the provisioning profile doesn't match certain criteria, your app won't launch.

Note:

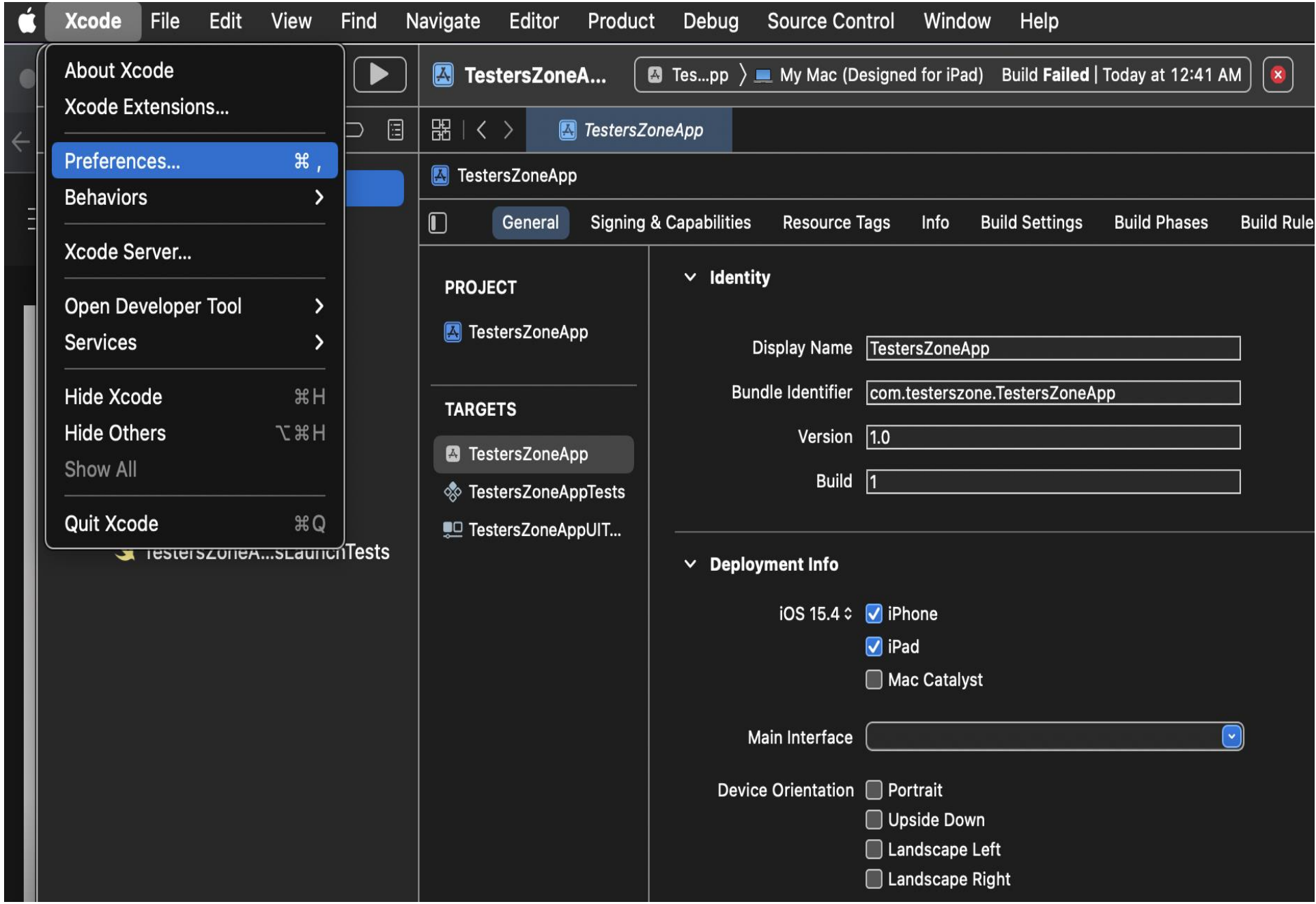
- Every iOS app's have to go through code signing and provisioning profile before it can be installed in iOS device even for web driver agent also it applicable before getting installed in iOS device. Appium handles this requirement in three different ways.

  A. Basic Automatic Configuration.
  B. Basic Manual Configuration.
  C. Full Manual Configuration.

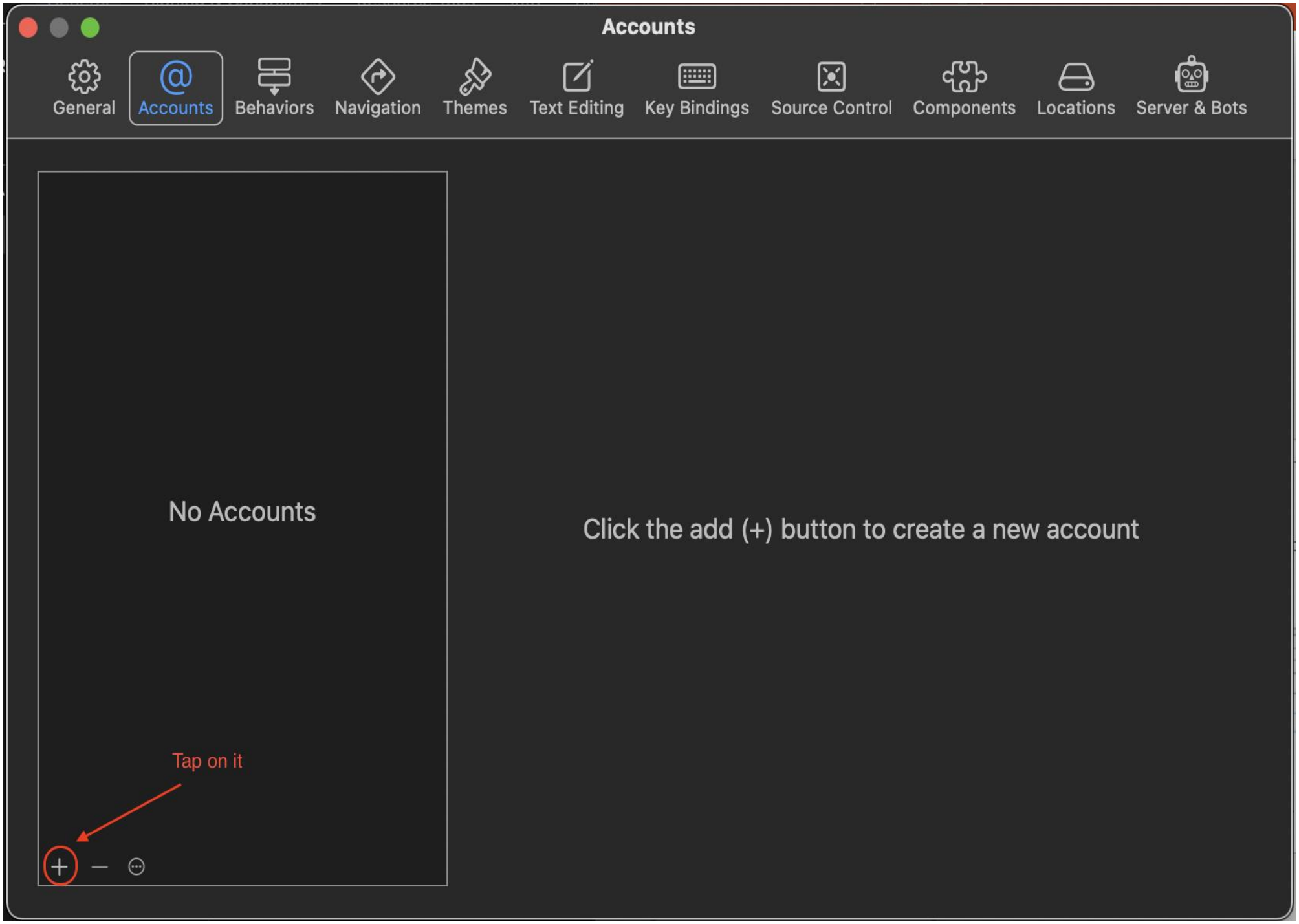
- Basic Manual configuration works well with developer paid account
- If you don't have any paid account and you are in learning phase better to use Full Manual Configuration.

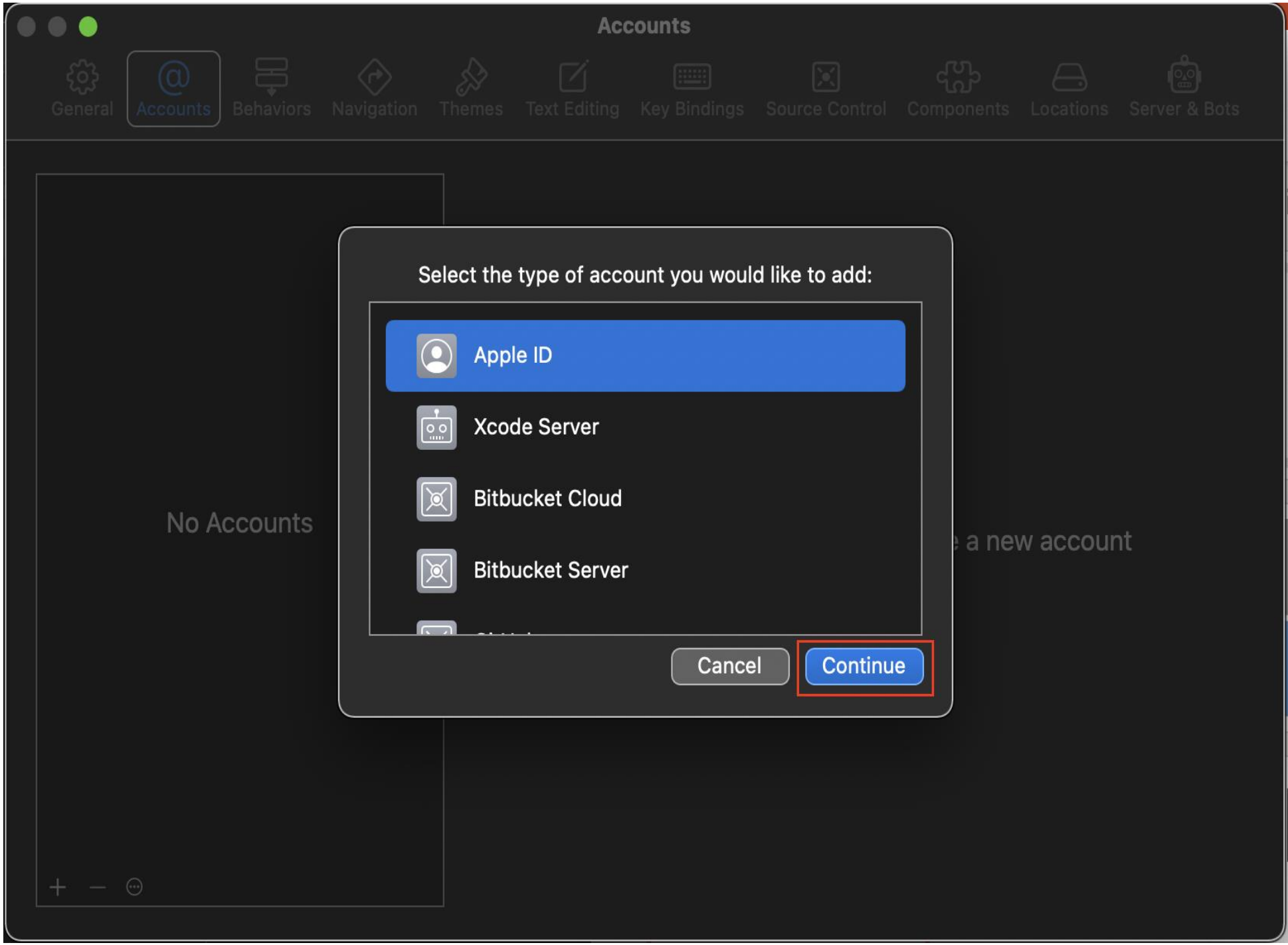
Full Manual Configuration.

Steps:
1. Open XCode project.
2. Click on XCode--> Preferences.



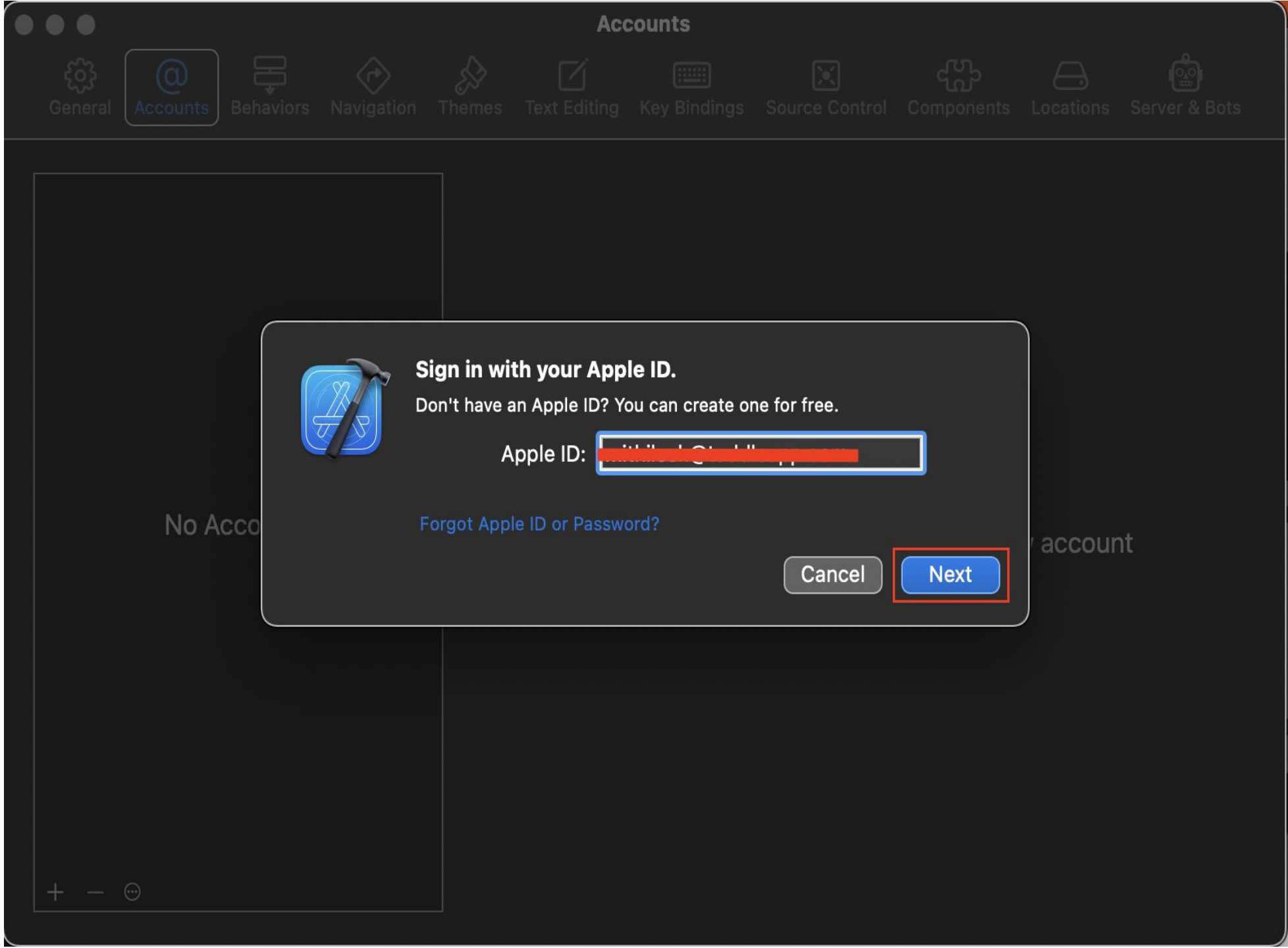
Click on + icon

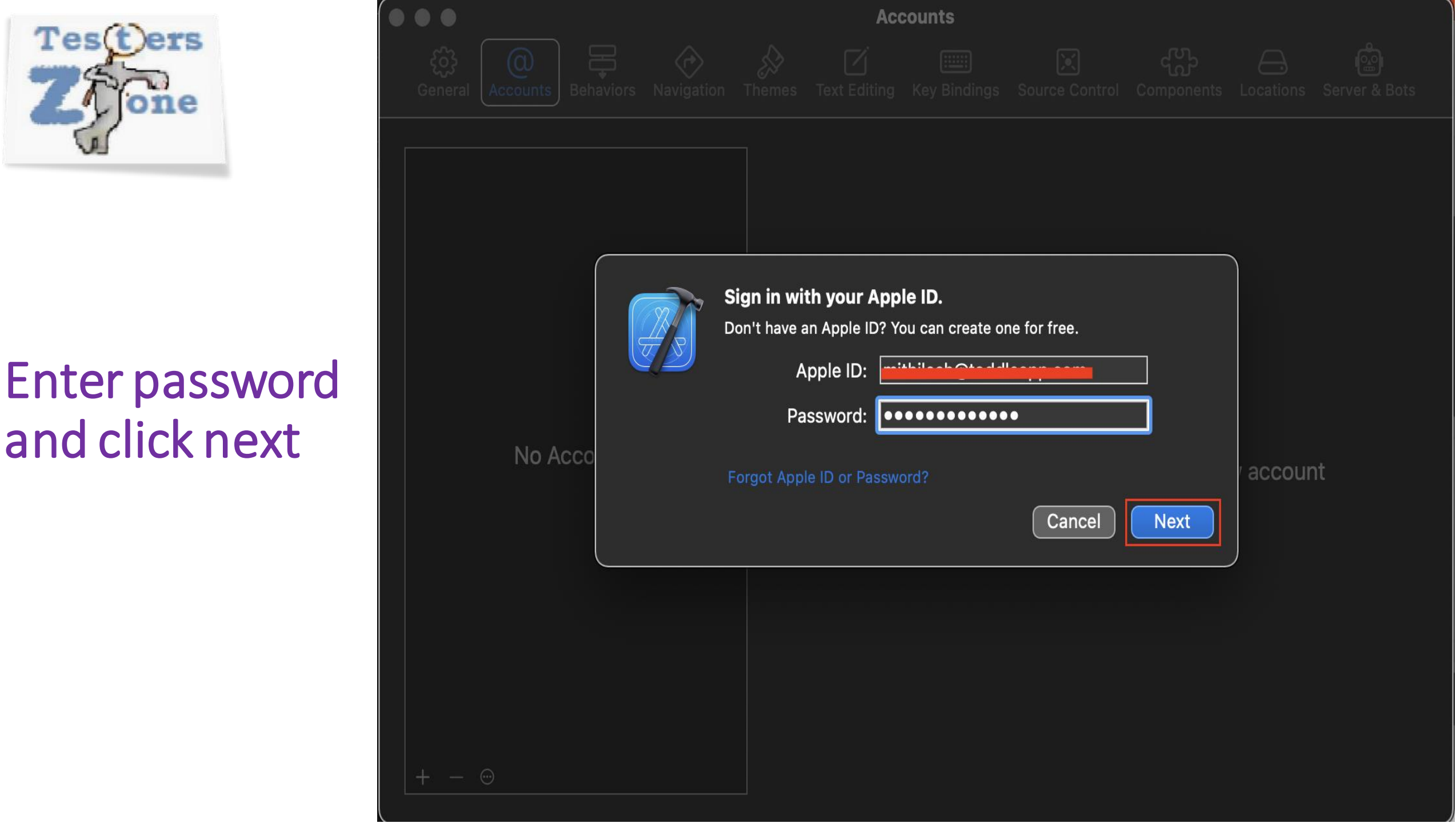




Select Apple ID and click on Continue

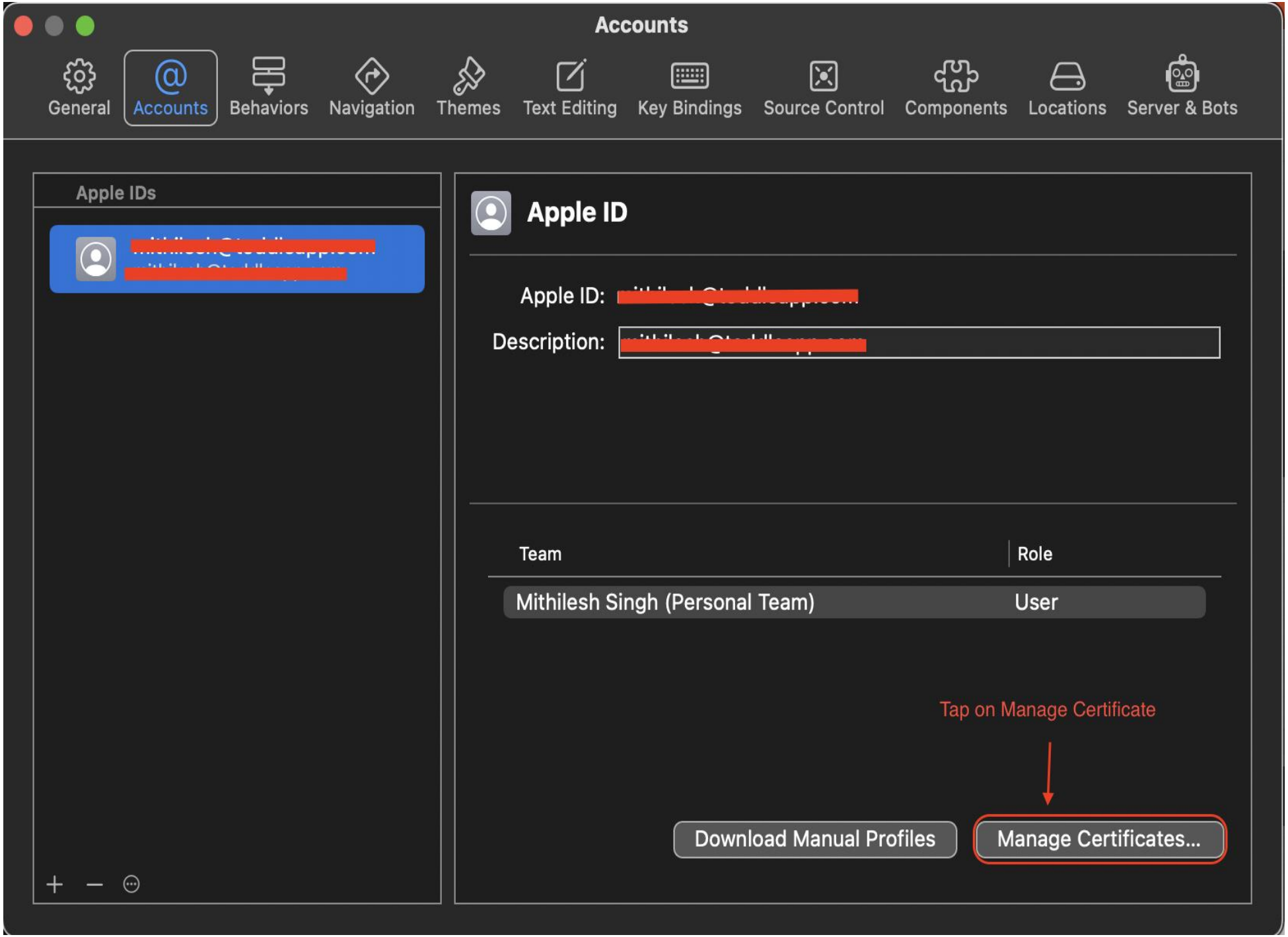
Enter your apple id and click next



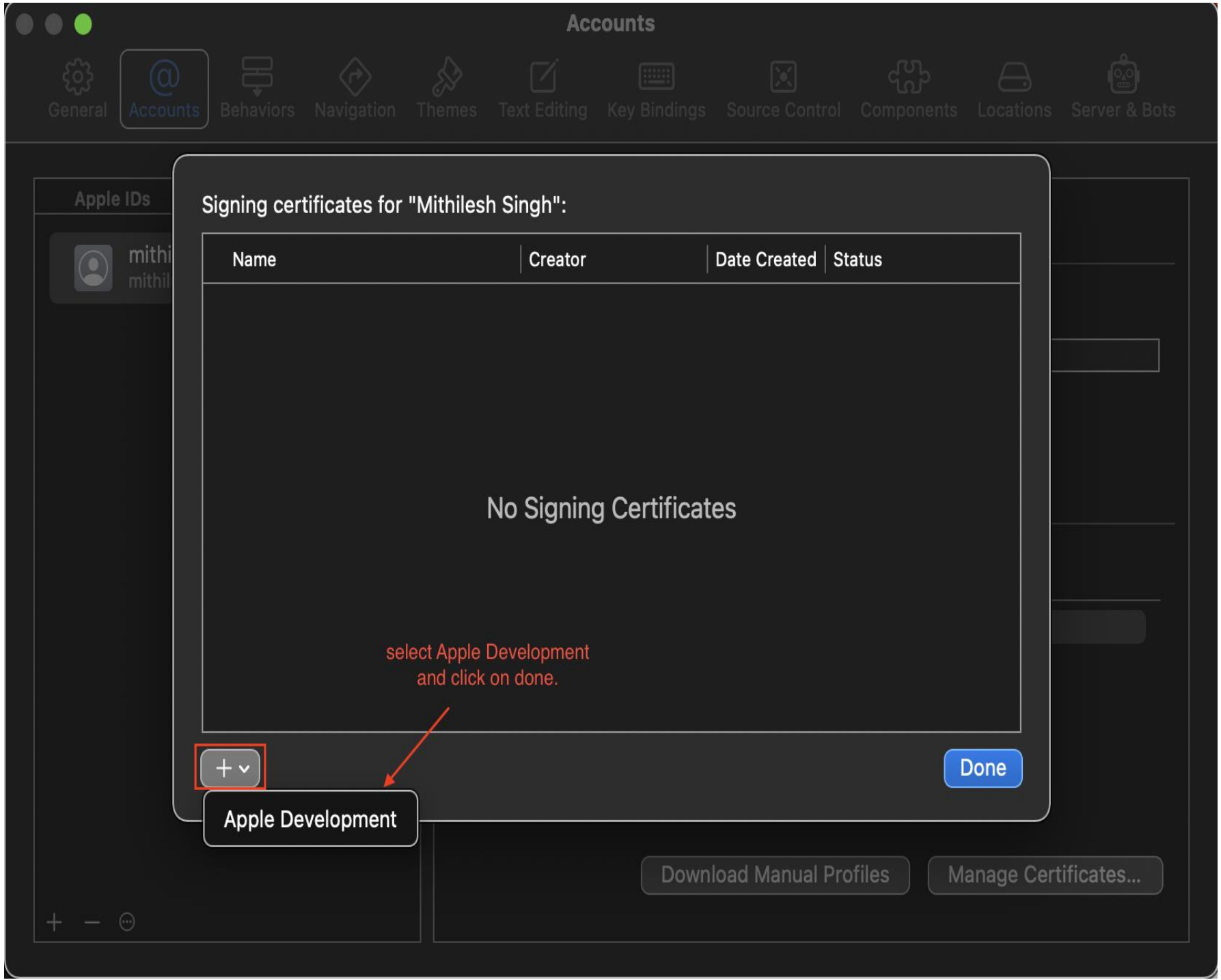


Enter password and click next

Click on Manage Certificates



Click on + icon and select Apple Development and click done button



1. Open the terminal and navigate to the appium-webdriveragent project

If you have installed appium using npm then location of project would be:

/usr/local/lib/node_modules/appium/node_modules/appium-webdriveragent

If you have appium GUI then project location would be:

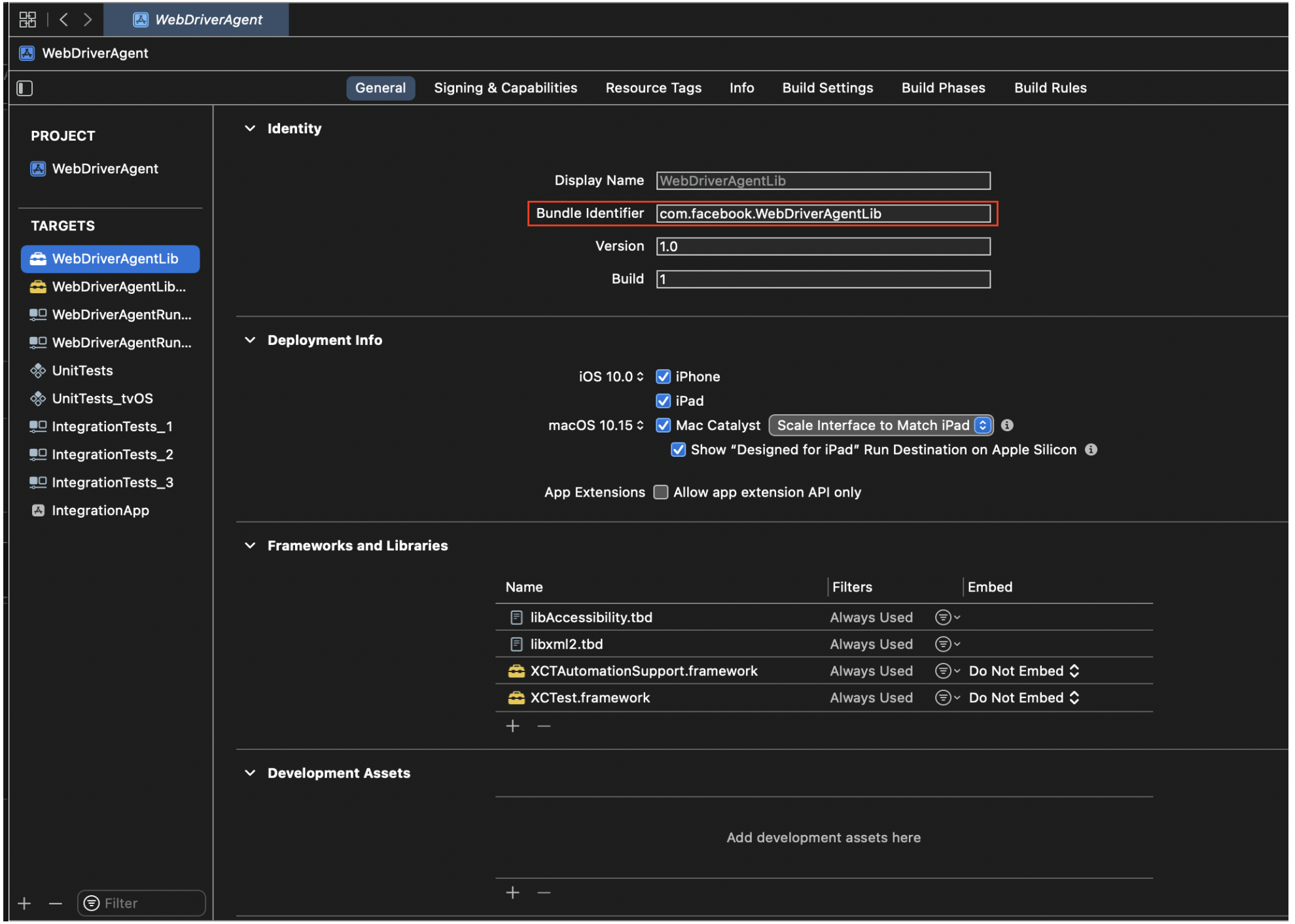
/Applications/Appium.app/Contents/Resources/app/node_modules/appium/node_modules/appium-webdriveragent

2. Use below command in the terminal after navigating to appium- webdriveragent folder.

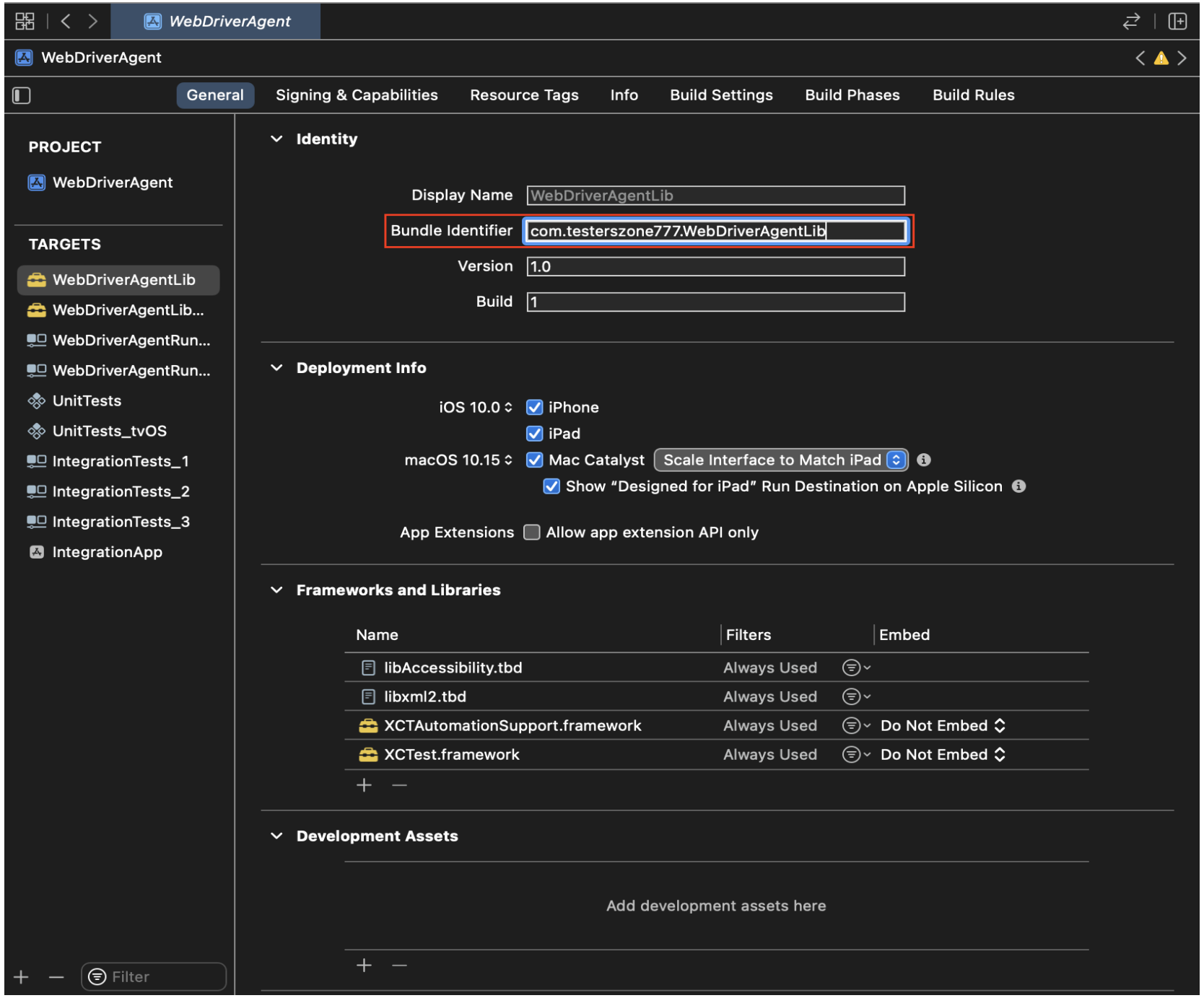
- mkdir –p Resources/WebDriverAgent.bundle

  ./Scripts/bootstrap.sh -d

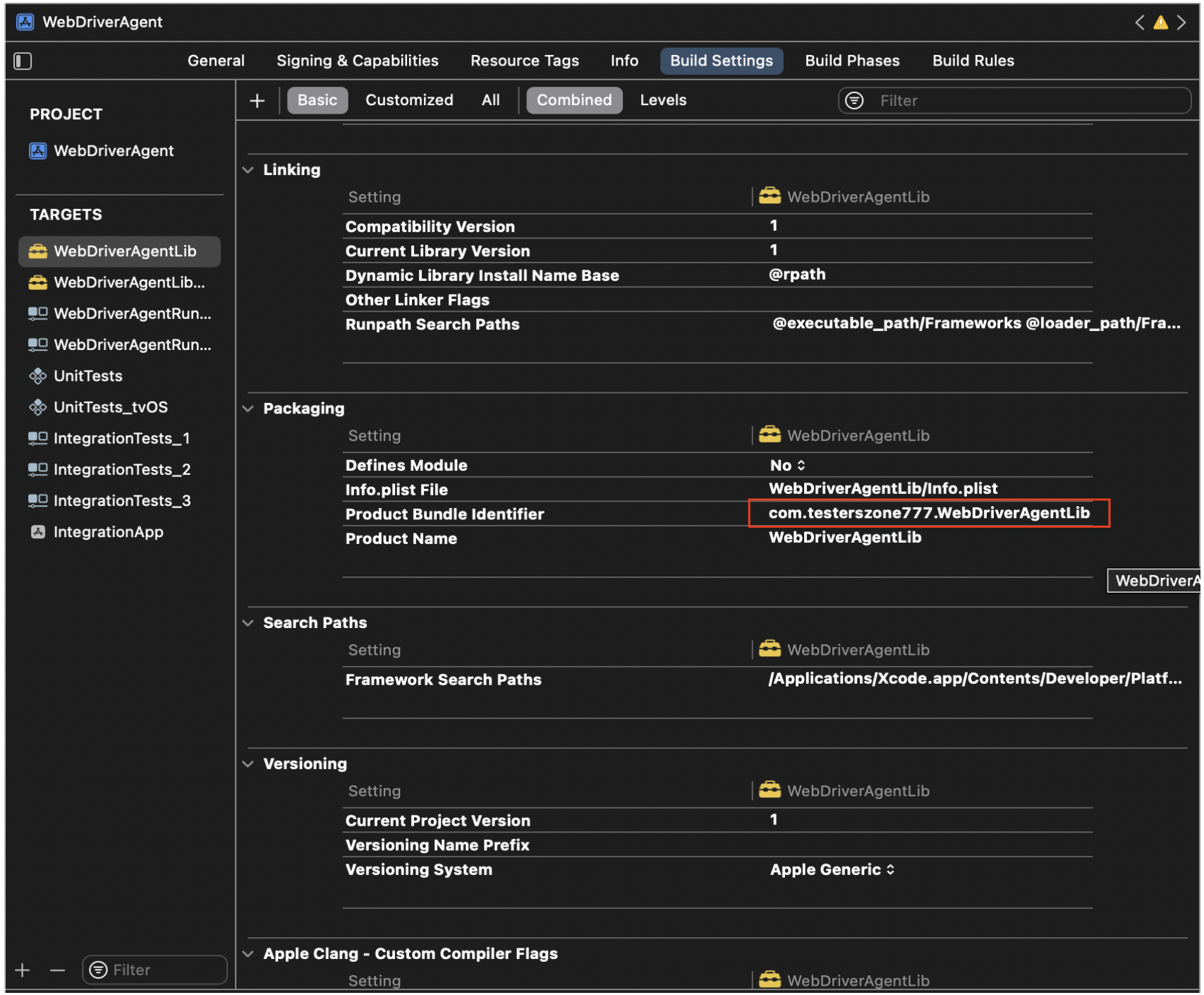
- 3. Navigate to the **appium-webdriveragent** folder and open project named "WebDriverAgent.xcodeproj"



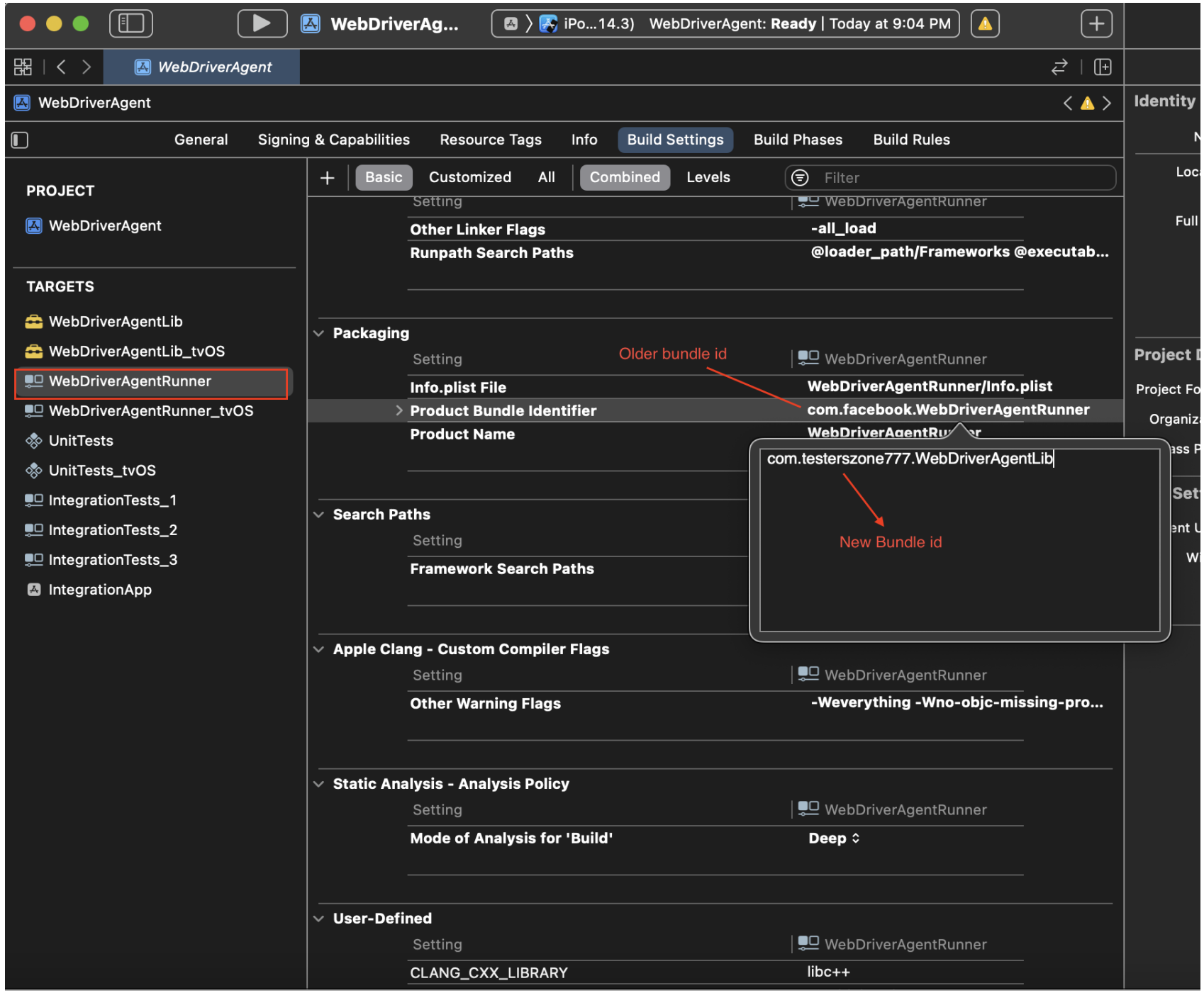
Update the new bundle id

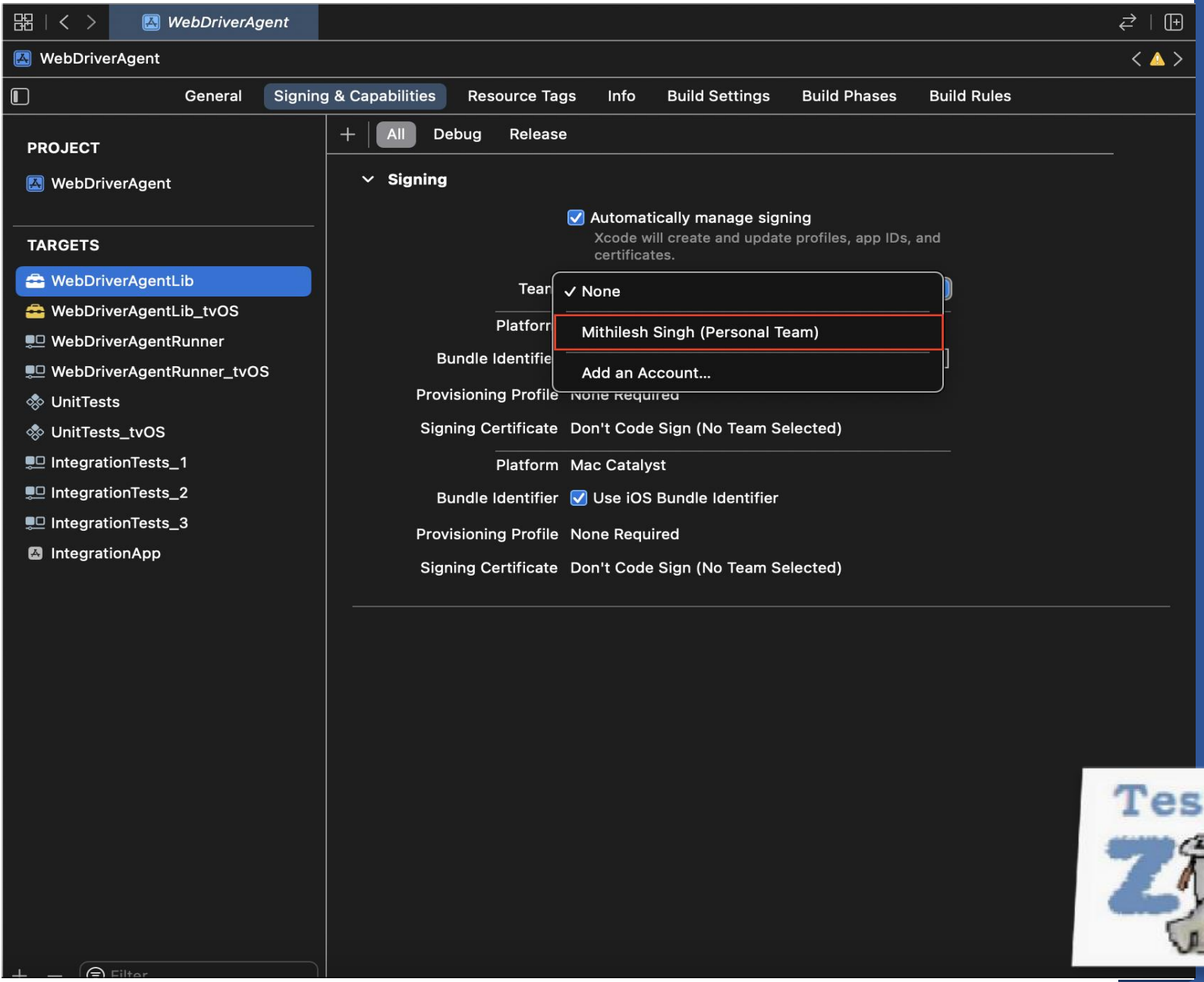


Verify same bundle id should be reflected under build settings



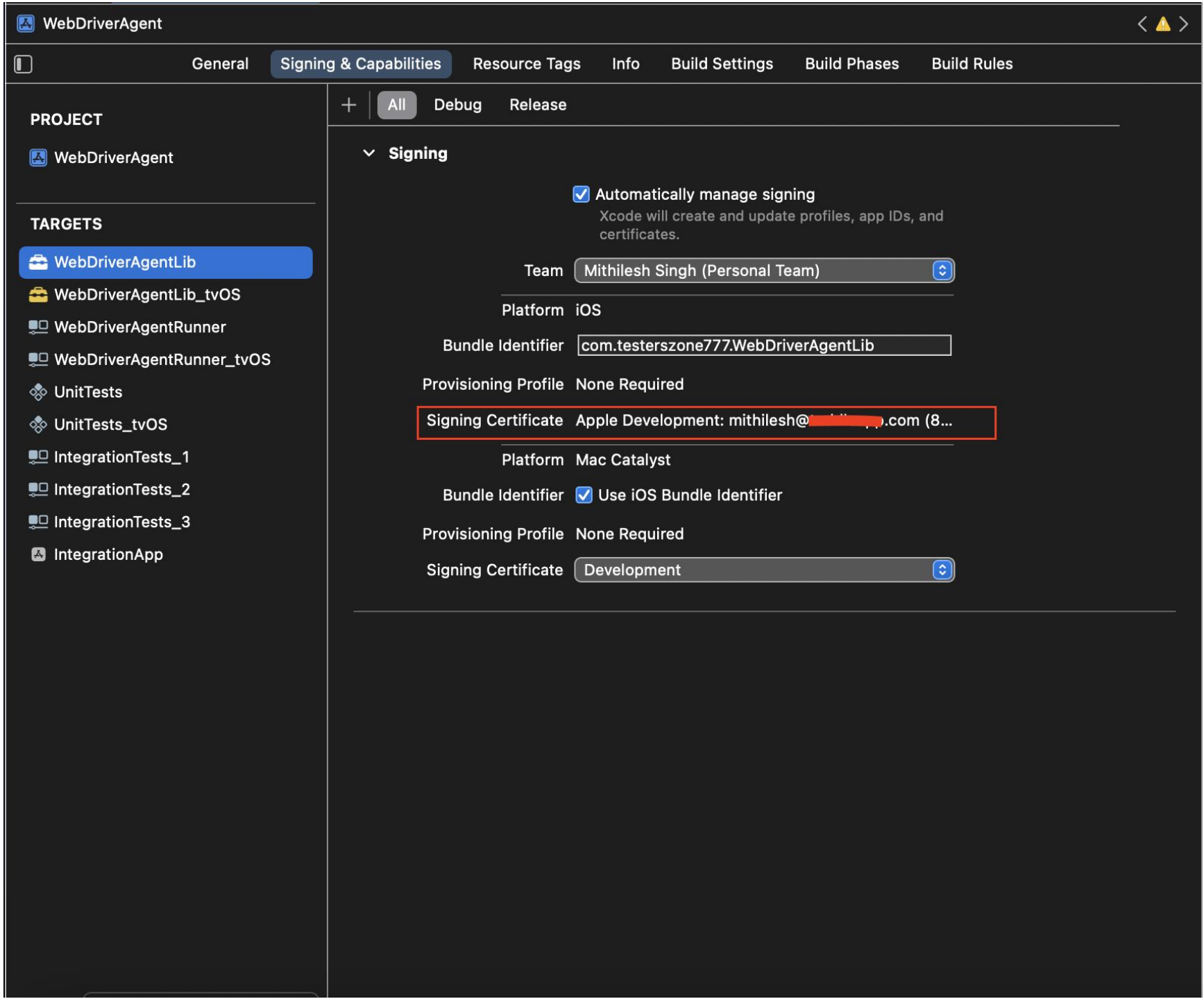
Modify bundle id under webDriver agent runner target as shown in screen shot





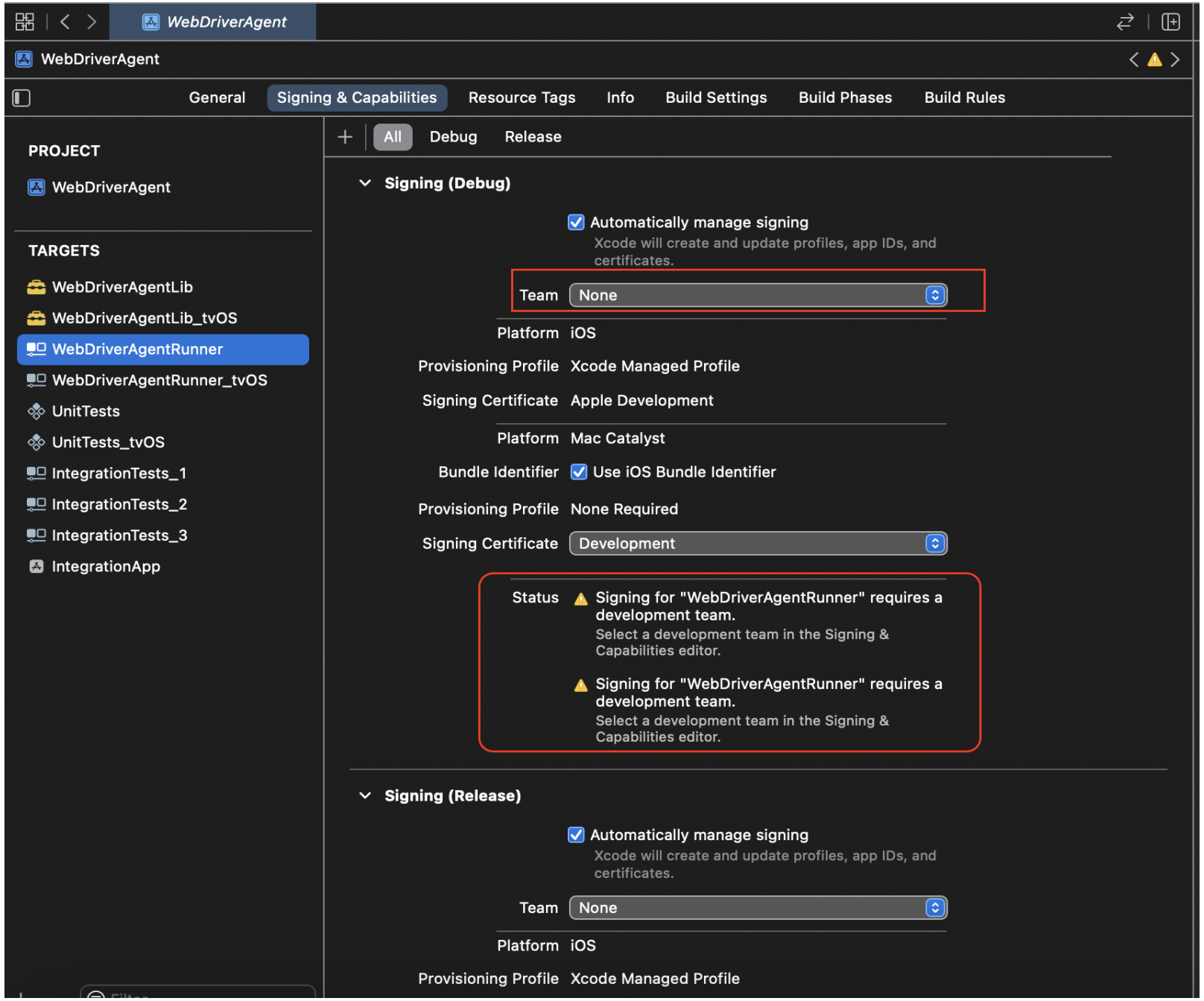
Select the team under web driver agent lib target--> signing and capabilities

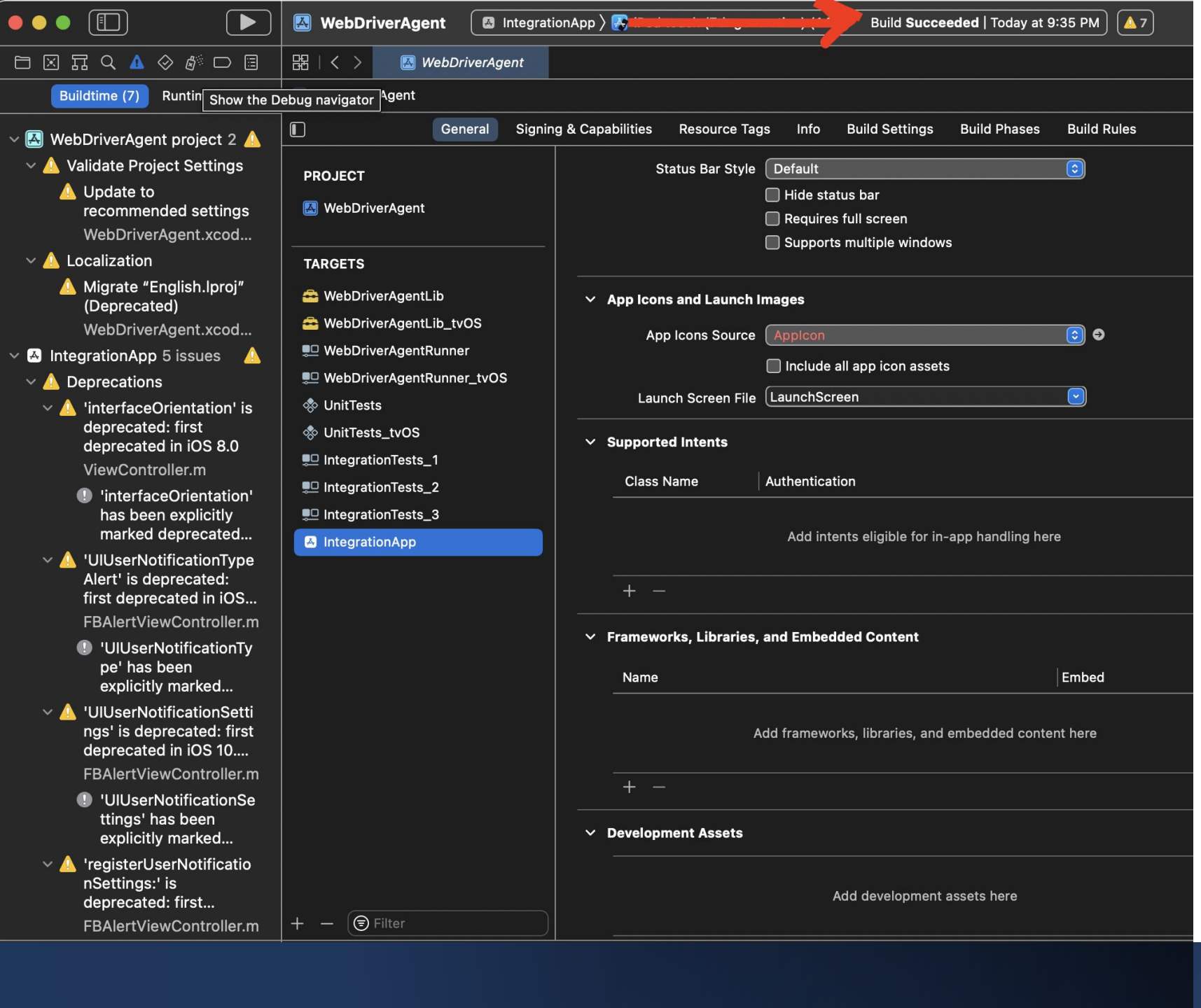
Once you will select the team, your signing certificate will get selected which we created.



Select team under **web driver agent runner --> Signing & Capabilities**

Note: you will observe warning until don't add team
once you will add team, it will use provisioning profile





Now all set to build the project. We can build either using play icon present at top left to the webDriverAgent as visible in the attachment or XCode--> Product--> Build.
Once after successful build you will be able to see Build Successful message on top as marked in the screenshot.

Note: if build is failed and error is there in integrationApp target, change bundle id name and add team and then build the project

Open the terminal--> navigate to the webDriverAgent folder accessing the path mentioned in earlier slide and execute below command:

```
xcodebuild project WebDriverAgent.xcodeproj -
 scheme WebDriverAgentRunner -destination 'id=<device UDID>' test
```

To get the device UDID we can run the command:
either 1. instruments –s devices or. 2. xcrun xctrace list devices

Note:

- If app is not getting launched check manually once, double click on installed webDriverAgent on iOS device, if you get untrusted developer pop up then you need to follow the below steps:

Go to Settings--> General-->Device Management--> click Trust <...>

Now you can use Appium and add desire capabilities mentioned in earlier slide and launch the app.

Note:

make sure your iPhone device is registered with provisioning profile which is used to create application under test. For that you can ask your developer to register your iPhone in the developer portal.

Appium set up on mac for android

Pre requisites:

- Install homebrew.
- Install node and npm.
- Install Appium server using NPM.
- Install Appium server using Appium Desktop client.
-

All these steps are already covered in "**Appium set up on mac for iOS**" in above slides..

Pre requisites:

- Install JAVA JDK
- Install Android Studio
- Set JAVA_HOME and ANDROID_STUDIO in the .bash_profile or .zshrc file.
- Verify installation using **appium- doctor**--> **Covered in Slide no. 14**

Install JAVA JDK

- First check if java is installed already in the system or not?
  Use command: **java –version**
- **I**f it is installed you will get the jdk version.
- Or else you can install it from download
  link: https://www.oracle.com/java/technologies/downloads/#java18

  Java installation for a mac might ask the oracle login credential so if you don't have then create one oracle account and use that.
- You need to select the specific .dmg java file which needs to be installed.
- Once it is downloaded, double click on jdk and install it in system.

Install Android Studio

- We need android sdk as per Appium documentation. Sdk comes as bundle with Android studio so either we can install only android sdk or android studio.
- Download link: https://developer.android.com/studio
- Once it is downloaded, double click on that and do install in your system.
- Move android studio to the App directory using drag and drop
- Navigate to the app in application directory and right click on that and click on open. Click on **next --> next --> next** and provide apple password for sdk's installation.

Set JAVA_HOME and ANDROID_STUDIO in the .bash_profile or .zshrc file

- Navigate to home directory: cd ~/
- Open either bash profile or zshrc file and the paths.
  Command to open bash profile : **open –e .bash_profile**
  Command to open zshrc file : **open -e .zshrc**
  Note: if it is not created then use below command to create new one:"
  To create bash_profile : **touch .bash_profile**
  To create zshrc file : **touch .zshrc**
- Set below paths in any one of these files(either .bash_profile or zshrc)**:**

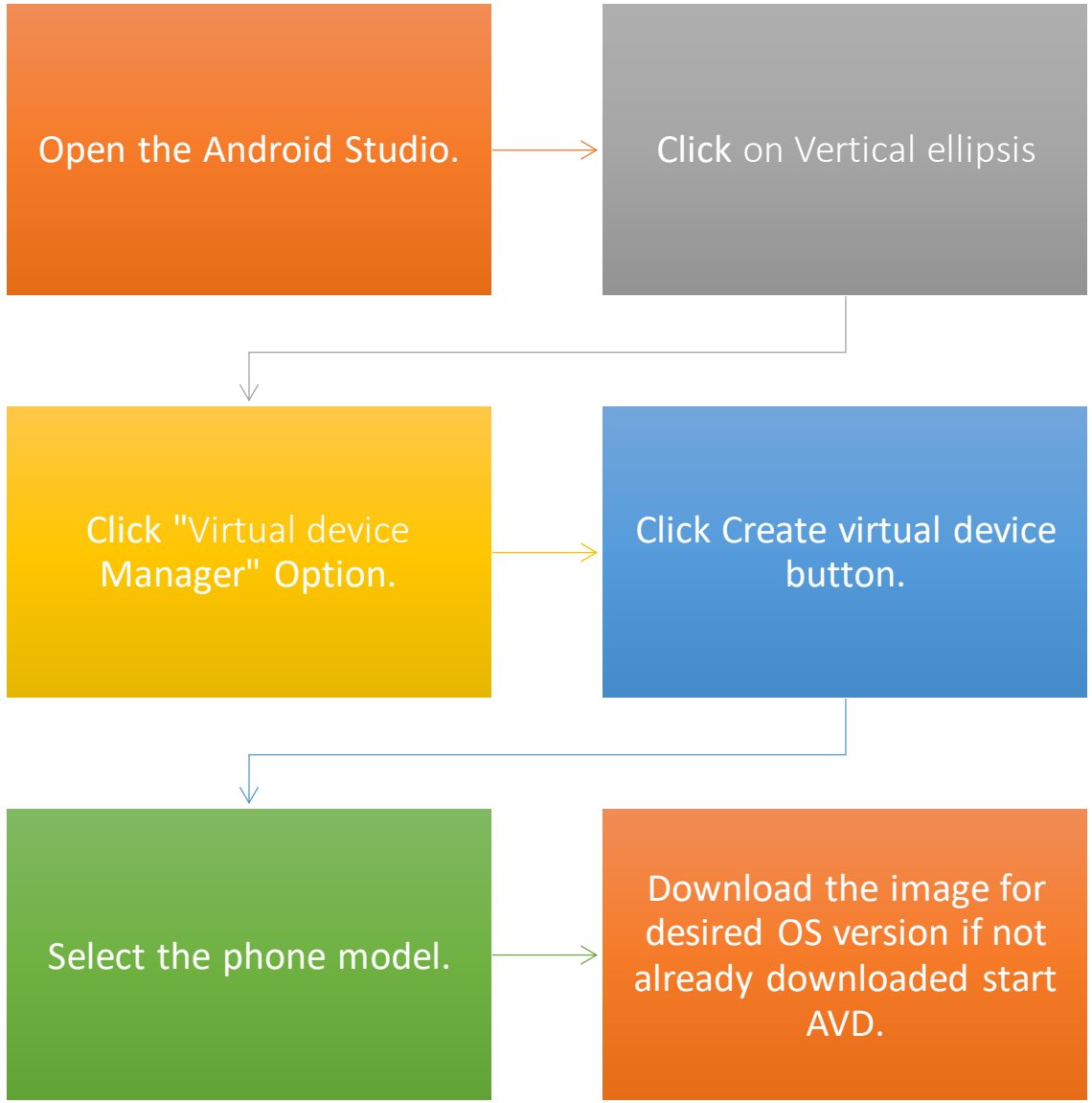
  1. export JAVA_HOME = $(/user/libexe/java_home)
  2. export ANDROID_HOME = $(HOME)/Library/Android/sdk
  3. export PATH= "${JAVA_HOME}/bin:${ANDROID_HOME}/tools:${ANDROID_HOME}/platform-tools:${PATH}"

Important points:

Changes we will make in .bash_profile or .zshrc file are not reflected automatically. For that we need to save those changes using command: **source .zshrc** or **source .bash_profile**

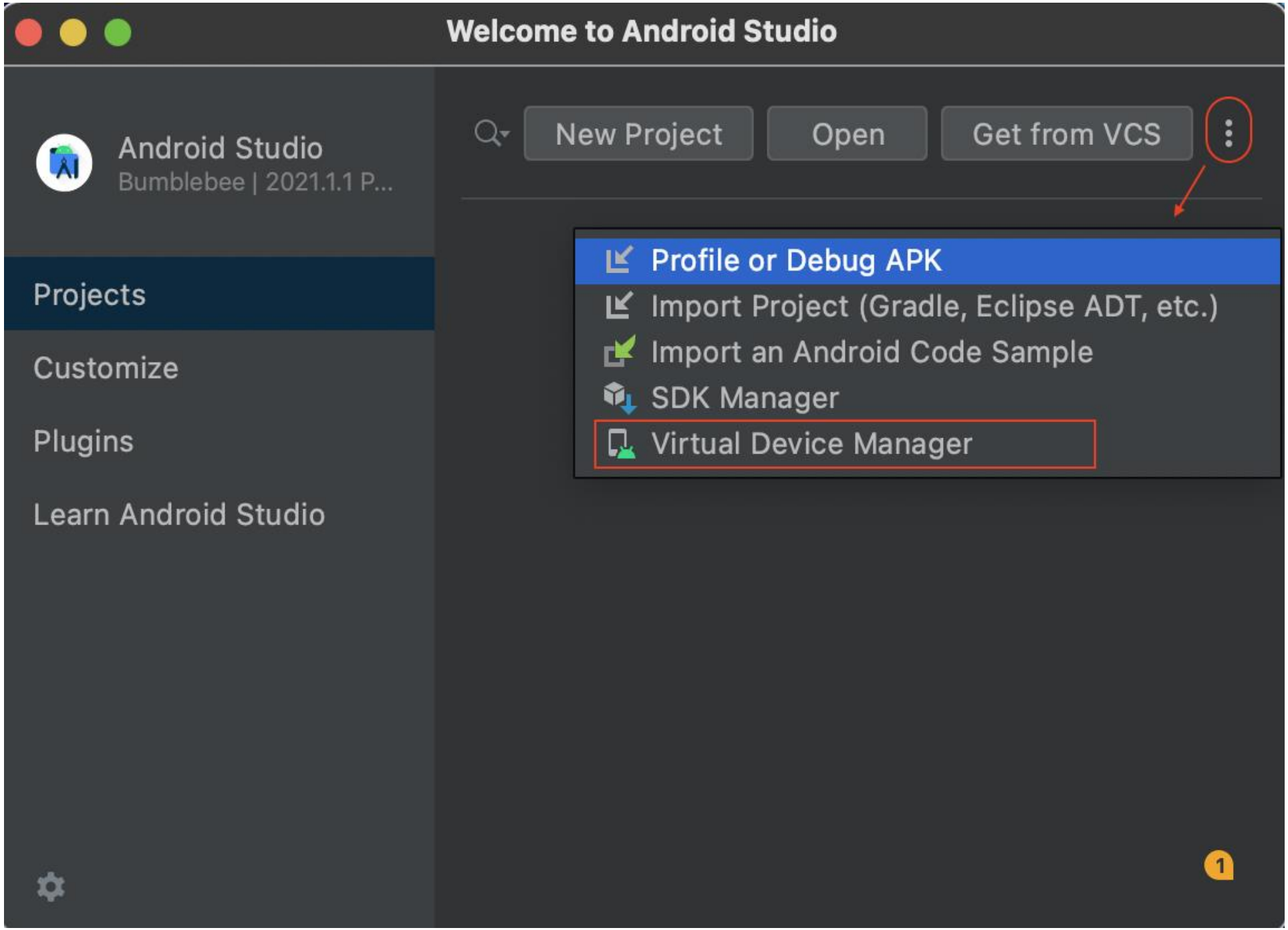
To get confirmed about our changes in the respective file we can use below command, it should return the exact path mentioned in the file**:**
echo $JAVA_HOME

Create Emulator/AVD(Android virtual device)

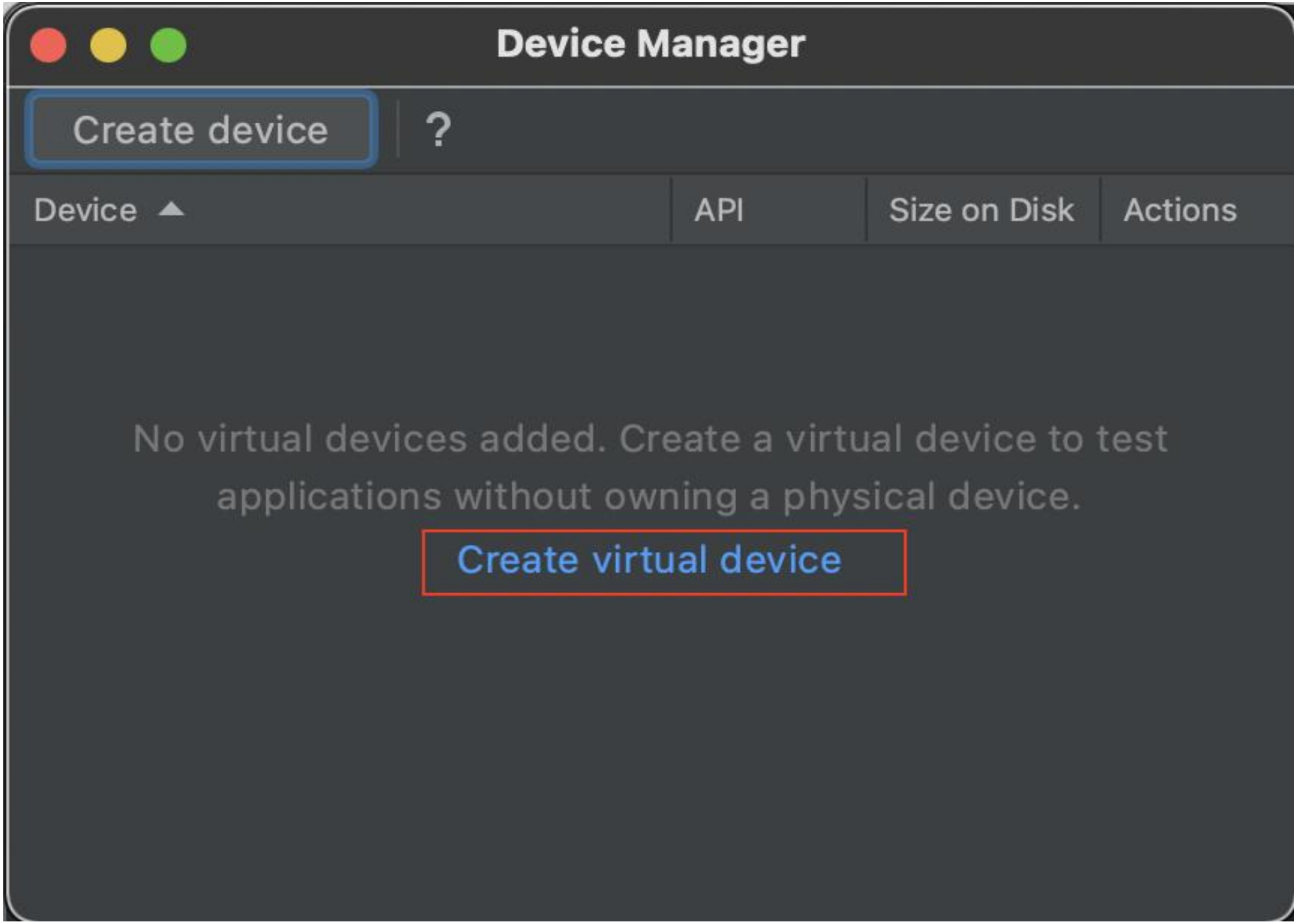


Testers Zone

Step: 1



Step: 2


Device Manager
Create device
?
Device
API
Size on Disk
Actions
No virtual devices added. Create a virtual device to test applications without owning a physical device.
Create virtual device

Select category as Phone and select any device e.g. Pixel 2 and click on next.

Virtual Device Configuration

Select Hardware

Choose a device definition

Category
TV
Phone
Wear OS
Tablet
Automotive

Name	Play St...	Size	Resolut...	Density
Pixel 3	▶	5.46"	108...	44...
Pixel 2 XL		5.99"	144...	56...
Pixel 2	▶	5.0"	108...	42...
Pixel	▶	5.0"	108...	42...
Nexus S		4.0"	480...	hdpi
Nexus One		3.7"	480...	hdpi

New Hardware Profile | Import Hardware Profiles

Pixel 2

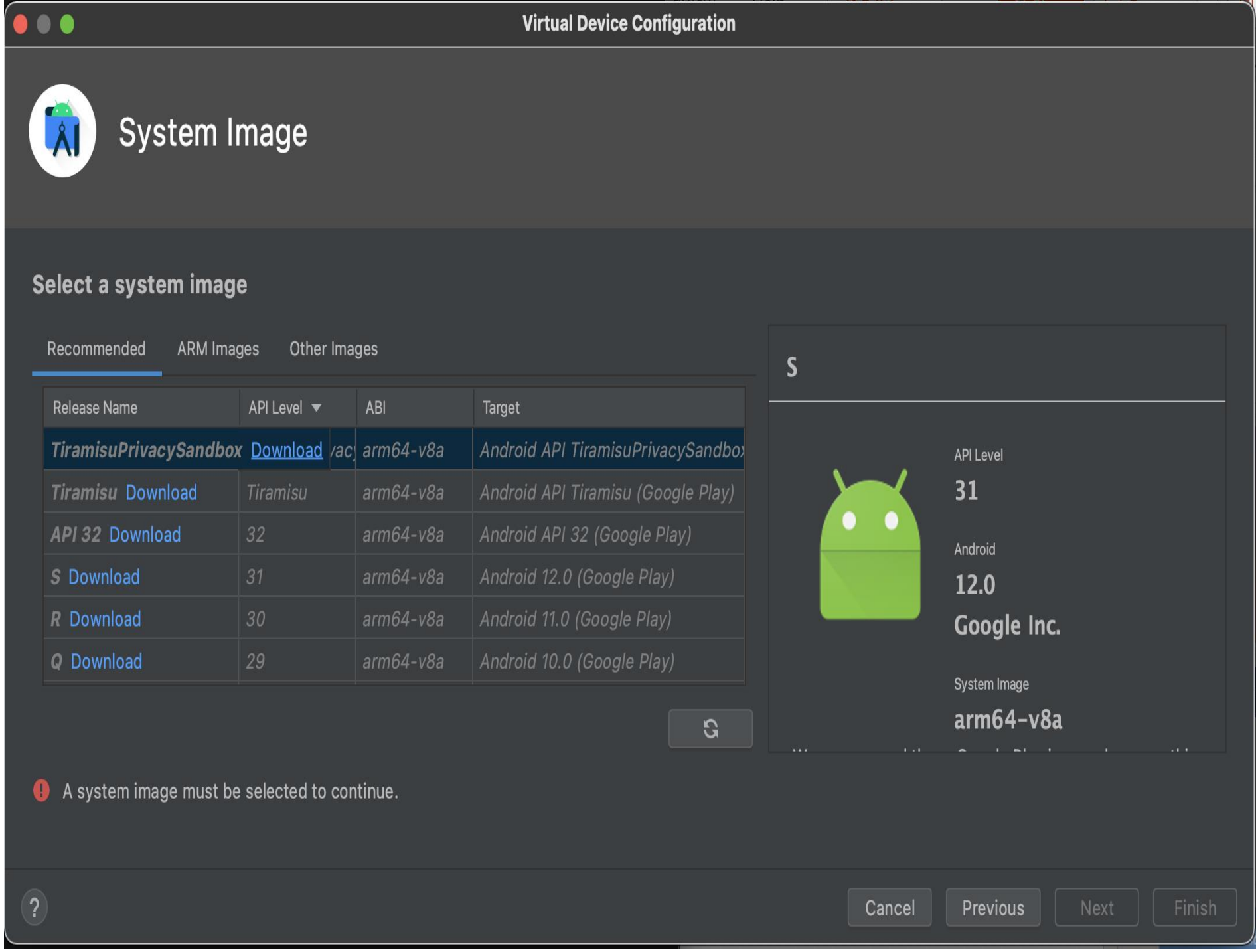
1080px
5.0"
1920px

Size: large
Ratio: long
Density: 420dpi

Clone Device...

Cancel | Previous | Next | Finish

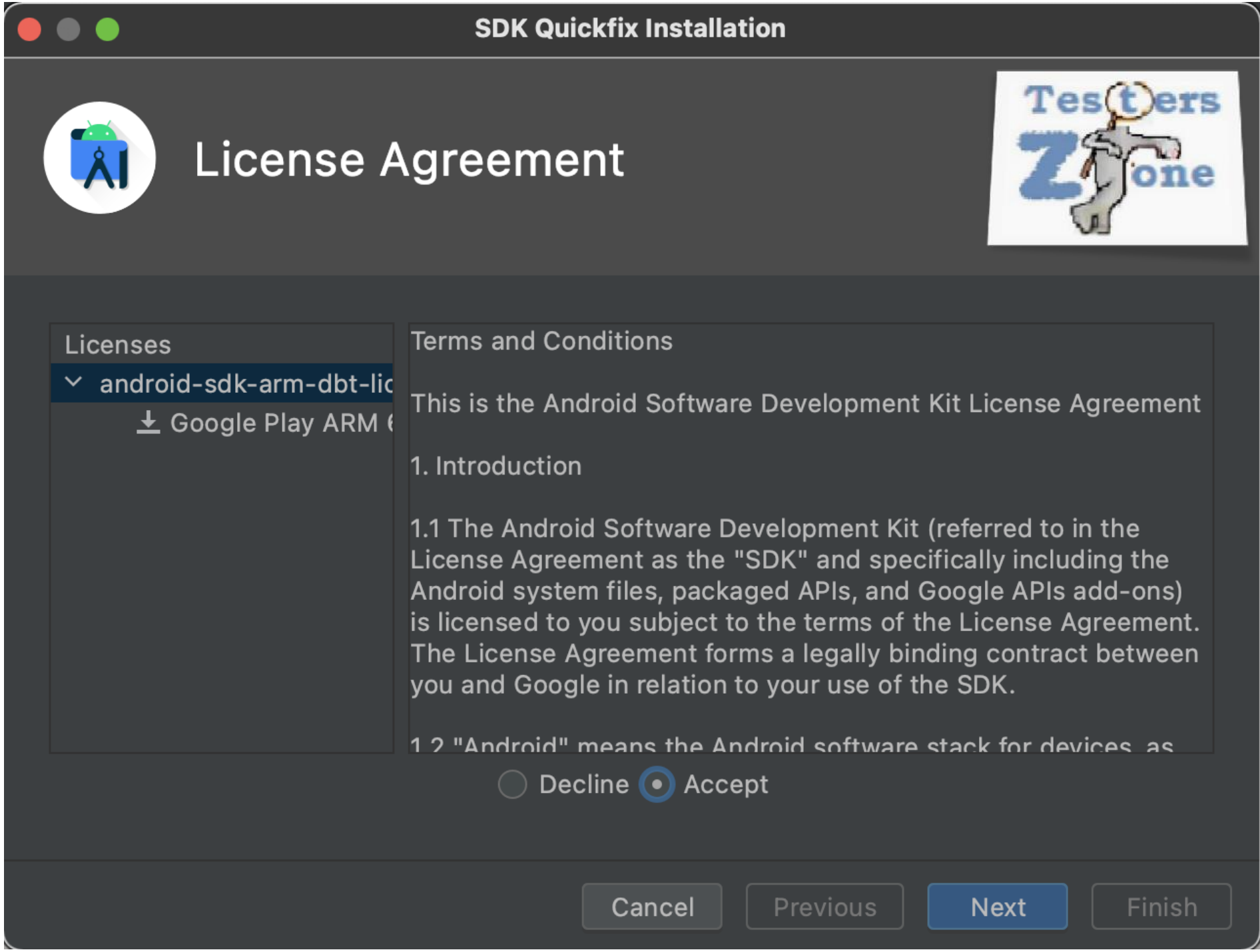
Click on Download for specific API level and download the System Image



Accept the license Agreement and click on next.

Note:

Now we have successfully created Emulator/ virtual device. We can launch it using play icon present in the action list.



Real device set up with dummy app

1. Download dummy app in your real device: https://github.com/appium/appium/blob/master/sample-code/apps/ApiDemos-debug.apk

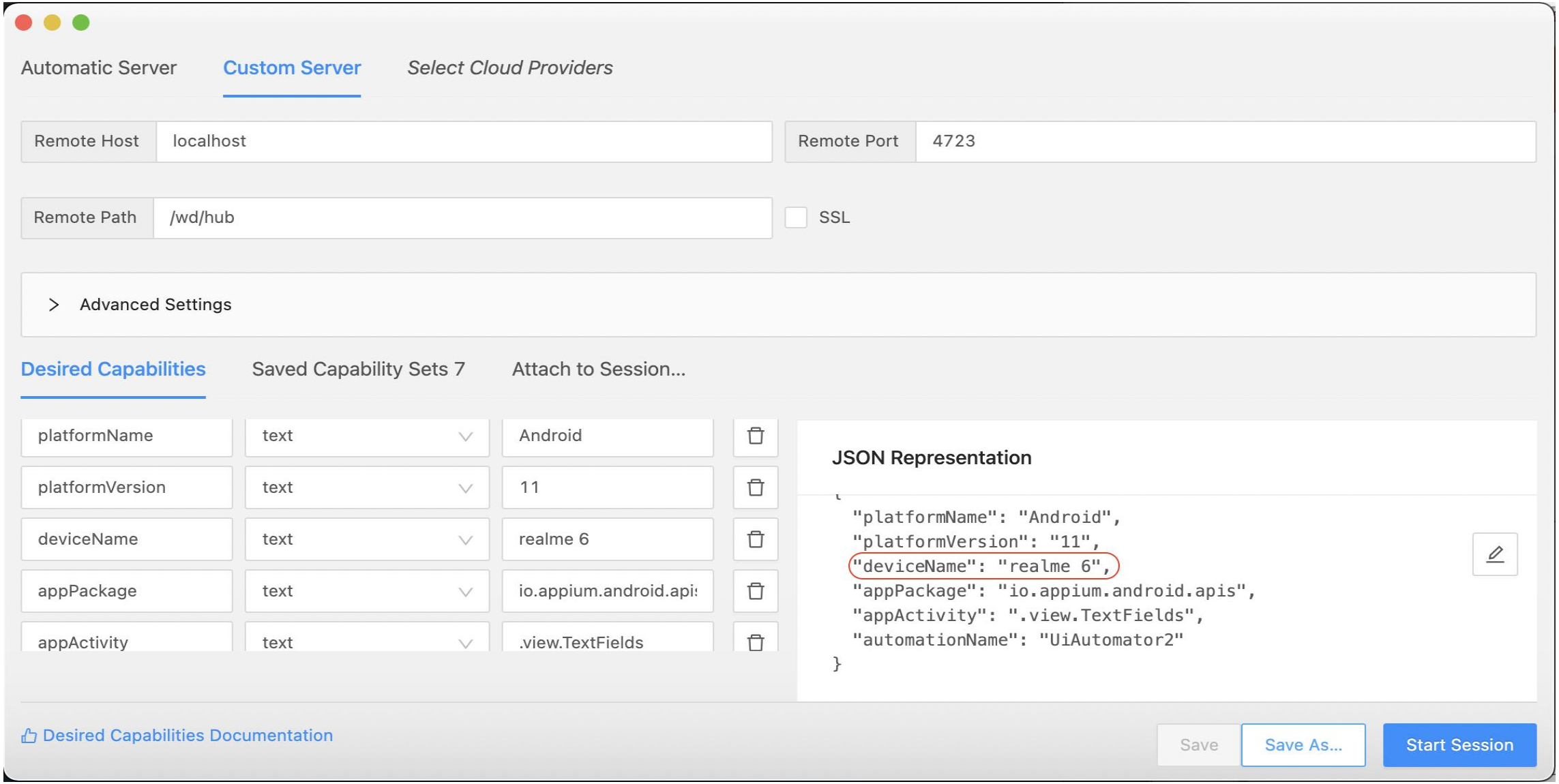
2. launch the Appium and start the server.

3. To launch the real device from appium we need to add desire capabilities.

4. Click on search icon present on the appium running server and click on desire capabilities.

5. add the capabilities: reference : http://appium.io/docs/en/about-appium/getting-started/?lang=en

Desire capabilities will be same for real and virtual device expect value of deviceName key



How to Enable USB debugging on Android[Real device]

There are following steps:

- Go to the "Settings" in your mobile.
- Click "About phone" option.
- Click on "Build Number" 7 to 8 times.
- Navigate back to "Settings".
- Open developer Options.
- Enable "USB Debugging"

Note:
once we enable USB Debugging, "Allow USB debugging" pop up will appear we need to click on checkbox and click Ok.

Note:

- Some time while launching the app using Appium desktop, might get error with adb tool because Appium Desktop cannot read ANDROID_HOME path from the bash profile.
- Set the ANDROID_HOME path in appium desktop using editing configuration.


Testers Zone
THANK YOU!