

---

# APPIUM SETUP

---

FOR MAC OSX

---

Chethan Shetty, ETSOM, ITS0

---

## Contents

1	PURPOSE .....	3
2	PRE-CONDITION .....	3
3	INSTALLATION .....	3
3.1	INSTALL JAVA DEVELOPEMENT KIT (JDK).....	3
3.2	INSTALL ANDROID STUDIO (v 3.1.4) .....	4
3.3	SETTING ENVIRONMENT VARIABLES .....	6
3.4	DOWNLOAD ECLIPSE IDE .....	6
3.5	MAVEN SETUP.....	7
3.6	MAVEN PLUGIN CONFIGURATION IN ECLIPSE.....	7
3.7	MAVEN DEPENDENCIES IN ECLIPSE .....	7
3.8	TESTNG PLUGIN CONFIGURATION IN ECLIPSE .....	8
3.9	REQUIRED INSTALLATION FOR APPIUM iOS MOBILE AUTOMATION .....	0
3.9.1	Homebrew Installation .....	0
3.9.2	Node.js Installation .....	1
3.9.3	Appium Installation.....	1
3.9.4	Installation and Settings of external dependencies .....	1
3.9.5	Development certificate and Provisioning profile: .....	2
	.....	2
3.9.6	WebDriverAgent .....	5
3.10	Install Appium Desktop:.....	7
3.11	Element Inspection for iOS and Android .....	9

## 1 PURPOSE

This document has the steps for complete set up of Appium in MAC OSX to test iOS and Android real devices.

## 2 PRE-CONDITION

1. MAC OSX Version – 10.12.x or higher version (MAC OS SIERRA)
2. XCODE version – 9.x or higher (Available in APP store- Latest xocde is recommended)

## 3 INSTALLATION

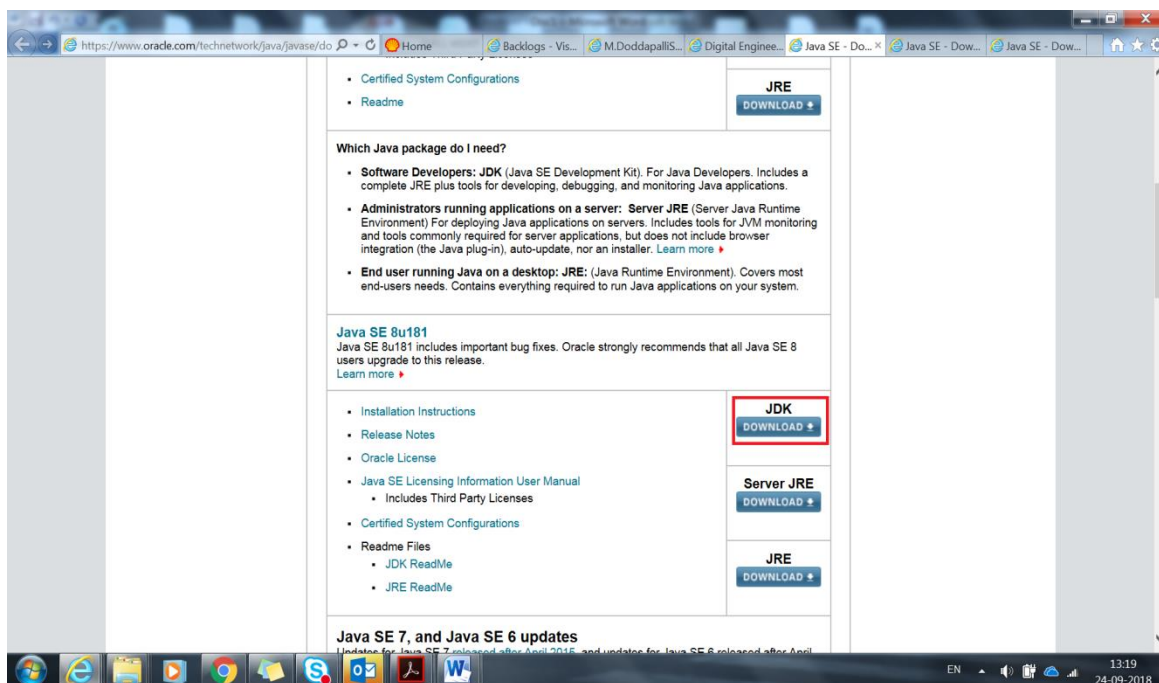
### 3.1 INSTALL JAVA DEVELOPEMENT KIT (JDK)

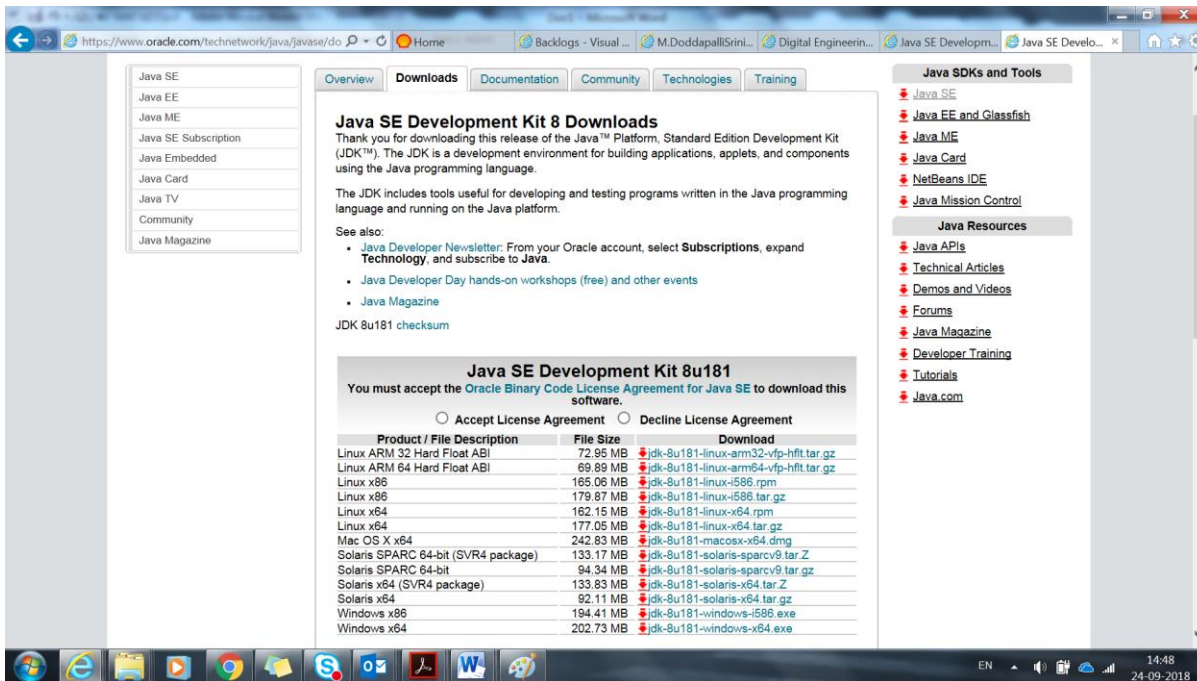
a. Download and install JDK from the following site

<https://www.oracle.com/technetwork/java/javase/downloads/index.html>

b. Select 'Accept License Agreement' and Click Download link corresponding to Mac OS X.

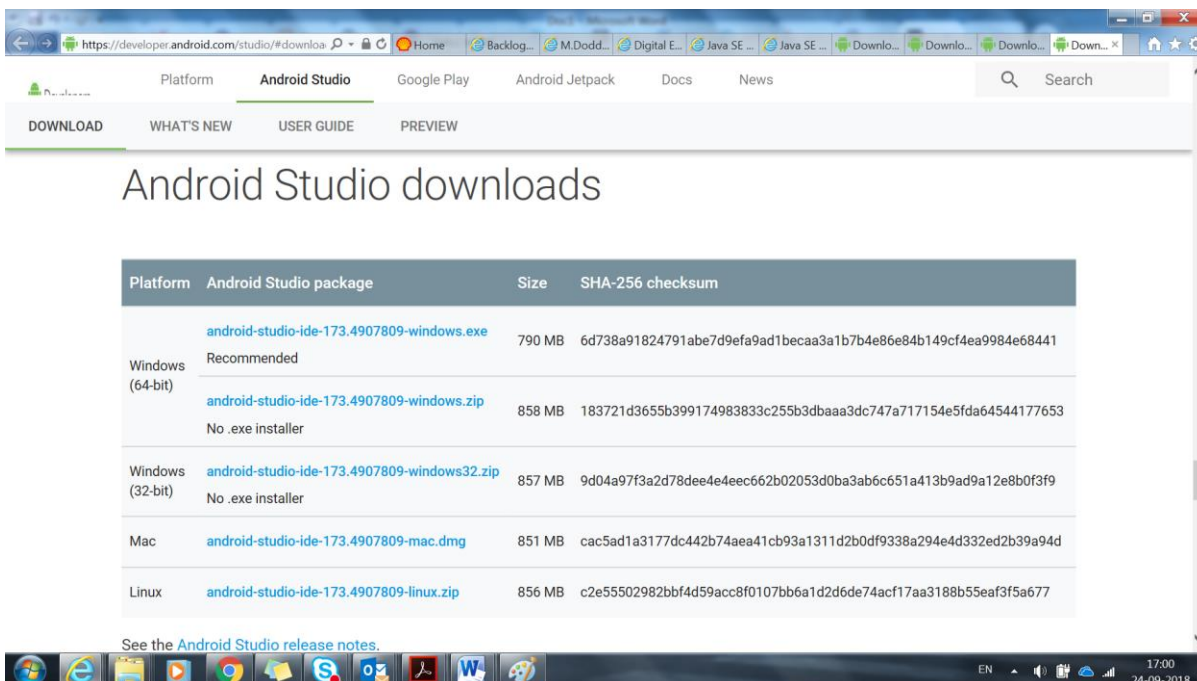
Note: Appium full-fledged framework uses a lot of dependencies. And not all these dependencies might work correctly with Java 10. **Hence, for Java, recommendation is to update to the most recent version of Java 8, so that you don't face any compatibility issues due to Java version 9 or 10.**





## 3.2 INSTALL ANDROID STUDIO (v 3.1.4)

- Open <https://developer.android.com/studio/#downloads>
- Download "Android Studio" for Mac.
- Run the downloaded installer. Follow the on-screen instruction and accept the defaults to complete the installation. Take note on the installation locations of "Android Studio" and "Android SDK".
- Android studio will be installed at the path: /Users/shell/Library/Android/sdk

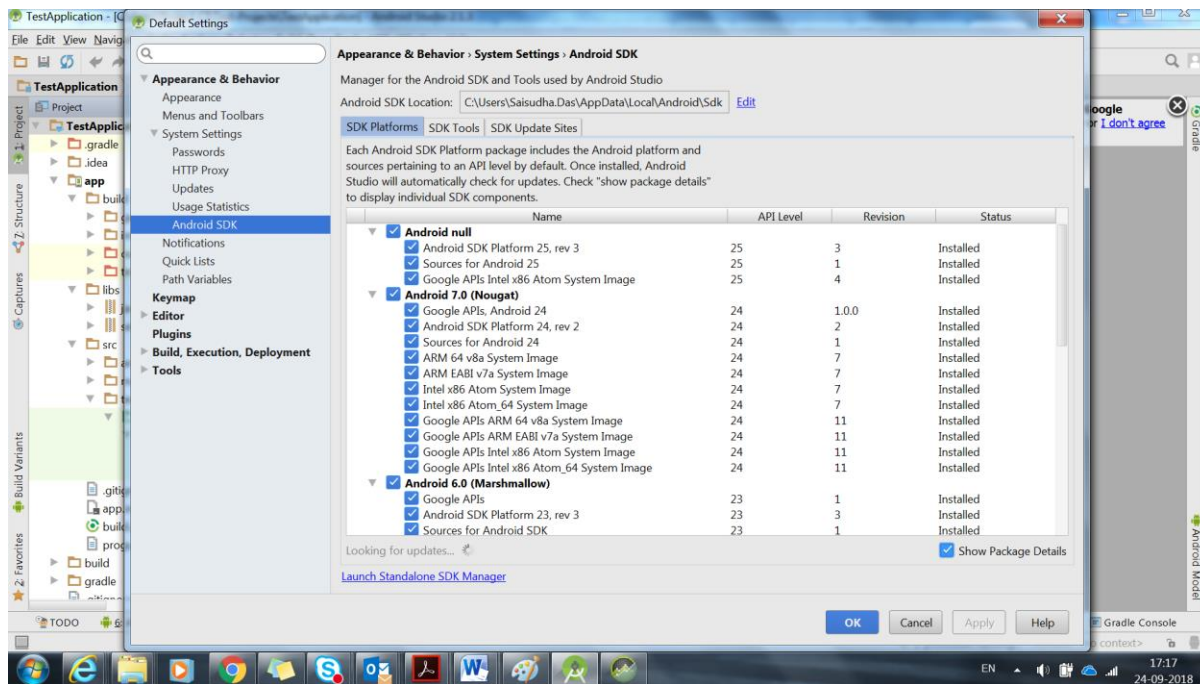


- e. Launch Android studio->Configure->SDK Manager

**NOTE:**

Please note that there are 3 main tabs in this screen – **SDK Platform**, **SDK Tools** and **SDK Update Sites**. We will be checking only the first two tabs – SDK Platform and SDK Tools

In **SDK Platform** tab, check that the latest version of Android is installed. Even if it shows Update available, it means that the version is installed. It's just that a new update is also available.



- f. Click on **SDK Tools** tab. Verify that the components highlighted in the red boxes in the below image are already installed.
- g. If all the components that are marked with red box in the above image are installed or in Update Available status, then you can click on OK button to close this screen. If any of these components is not installed, then you can click on the checkbox against that component and then click on Apply button to install the component.

Check that all the components that are in the red box are installed / updated

Name	Version	Status
Android SDK Build-Tools 28-rc2		Update Available: 28.0.0 rc2
GPU Debugging tools		Not Installed
CMake		Not Installed
LLDB		Not Installed
Android Auto API Simulators	1	Not installed
Android Auto Desktop Head Unit emulator	1.1	Not installed
Android Emulator	26.0.3	Update Available: 27.2.9
Android SDK Platform-Tools	27.0.1	Installed
Android SDK Tools	26.1.1	Installed
Documentation for Android SDK	1	Installed
Google Play APK Expansion library	1	Not installed
Google Play Licensing Library	1	Not installed
Google Play services	49	Not installed
Google USB Driver	11	Not installed
Google Web Driver	2	Not installed
Instant Apps Development SDK	1.2.0	Not installed
Intel x86 Emulator Accelerator (HAXM installer)	6.2.1	Installed
NDK	17.0.4754217	Not installed
<b>Support Repository</b>		
ConstraintLayout for Android		Installed
Solver for ConstraintLayout		Installed
Android Support Repository	47.0.0	Installed
Google Repository	58	Installed

Show Package Details

### 3.3 SETTING ENVIRONMENT VARIABLES

Make sure you set the following home environment variables. Open a terminal window and go to your user folder and type in:

- Open bash profile by typing command `open .bash_profile`
- The mac text editor will open up. There you should include your android sdk and Java home paths.
- Add the below commands to the bash profile:

```
export JAVA_HOME=/Library/Java/JavaVirtualMachines/jdk1.8.0_181.jdk/Contents/Home
export PATH=$JAVA_HOME/bin:$PATH
export ANDROID_HOME=/Users/shell/library/Android/sdk (Replace shell with your
username)
export PATH=$ANDROID_HOME/platform-tools:$PATH
export PATH=$ANDROID_HOME/tools:$PATH
```

- Save the file using Command+S in terminal.

### 3.4 DOWNLOAD ECLIPSE IDE

Eclipse is an integrated development environment (IDE) and used most widely in Java IDE. It contains a base workspace and an extensible plug-in system for customizing the environment.

Eclipse is written in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including C, C++, JavaScript, Perl, PHP, Python, Ruby (including Ruby on Rails framework), etc.

- Download "Eclipse IDE for Java Developers" from <https://www.eclipse.org/downloads/packages/>

Eclipse IDE for Java Developers

189 MB 36,953 DOWNLOADS

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

Windows 32-bit 64-bit  
Mac Cocoa 64-bit  
Linux 32-bit 64-bit

- b. Select Mac CoCoa 64 bit for your laptop
- c. You should be able to download the latest version of eclipse (SimRel)

### 3.5 MAVEN SETUP

- a. Download MAVEN from <https://maven.apache.org/download.cgi>
- b. Apache Maven is a build management tool used for Java projects.

	Link	Checksums	Signature
Binary tar.gz archive	<a href="#">apache-maven-3.5.4-bin.tar.gz</a>	<a href="#">apache-maven-3.5.4-bin.tar.gz.sha512</a>	<a href="#">apache-maven-3.5.4-bin.tar.gz.asc</a>
Binary zip archive	<a href="#">apache-maven-3.5.4-bin.zip</a>	<a href="#">apache-maven-3.5.4-bin.zip.sha512</a>	<a href="#">apache-maven-3.5.4-bin.zip.asc</a>
Source tar.gz archive	<a href="#">apache-maven-3.5.4-src.tar.gz</a>	<a href="#">apache-maven-3.5.4-src.tar.gz.sha512</a>	<a href="#">apache-maven-3.5.4-src.tar.gz.asc</a>
Source zip archive	<a href="#">apache-maven-3.5.4-src.zip</a>	<a href="#">apache-maven-3.5.4-src.zip.sha512</a>	<a href="#">apache-maven-3.5.4-src.zip.asc</a>

- c. You should download the "bin.zip" file highlighted above and save it on your system. You can extract the "apache-maven-3.5.4" folder anywhere you want in your PC.
- d. Next – Set the path in the bash profile:
- e. Open bash profile. Enter "nano ~/.bash\_profile" in terminal and click return
- f. Add the below lines to the bash profile

```
Export MAVEN_HOME= Users/shell/Documents/apache-maven-3.5.4
Export PATH=$MAVEN_HOME:$PATH
```

- g. Save the changes in the bash profile and exit.

### 3.6 MAVEN PLUGIN CONFIGURATION IN ECLIPSE

Apache Maven is the most widely used build tool for Java-based applications and beyond. Maven comes default with Neon and higher versions of Eclipse.

### 3.7 MAVEN DEPENDENCIES IN ECLIPSE

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. All the jars respectively for Selenium, appium java client, xml, apache poi etc. mentioned below is downloaded once we build the pom.xml in the Maven project in eclipse for Appium.

This is just for the information on the dependencies and jar file along with the versions.

```
<dependencies>
<!-- Selenium -->
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-java</artifactId>
<version>3.11</version>
</dependency>
```

```
<!-- java client -->
<dependency>
<groupId>io.appium</groupId>
<artifactId>java-client</artifactId>
<version>6.0.0-BETA 4</version>
```

```
</dependency>
```

```
<!-- testng -->  
<dependency>  
<groupId>org.testng</groupId>  
<artifactId>testng</artifactId>  
<version>6.10</version>  
</dependency>
```

```
<!-- apache POI for excel -->  
<dependency>  
<groupId>org.apache.poi</groupId>  
<artifactId>poi</artifactId>  
<version>3.10-FINAL</version>  
<type>jar</type>  
</dependency>
```

```
<dependency>  
<groupId>org.apache.poi</groupId>  
<artifactId>poi-ooxml</artifactId>  
<version>3.10-FINAL</version>  
<type>jar</type>  
</dependency>
```

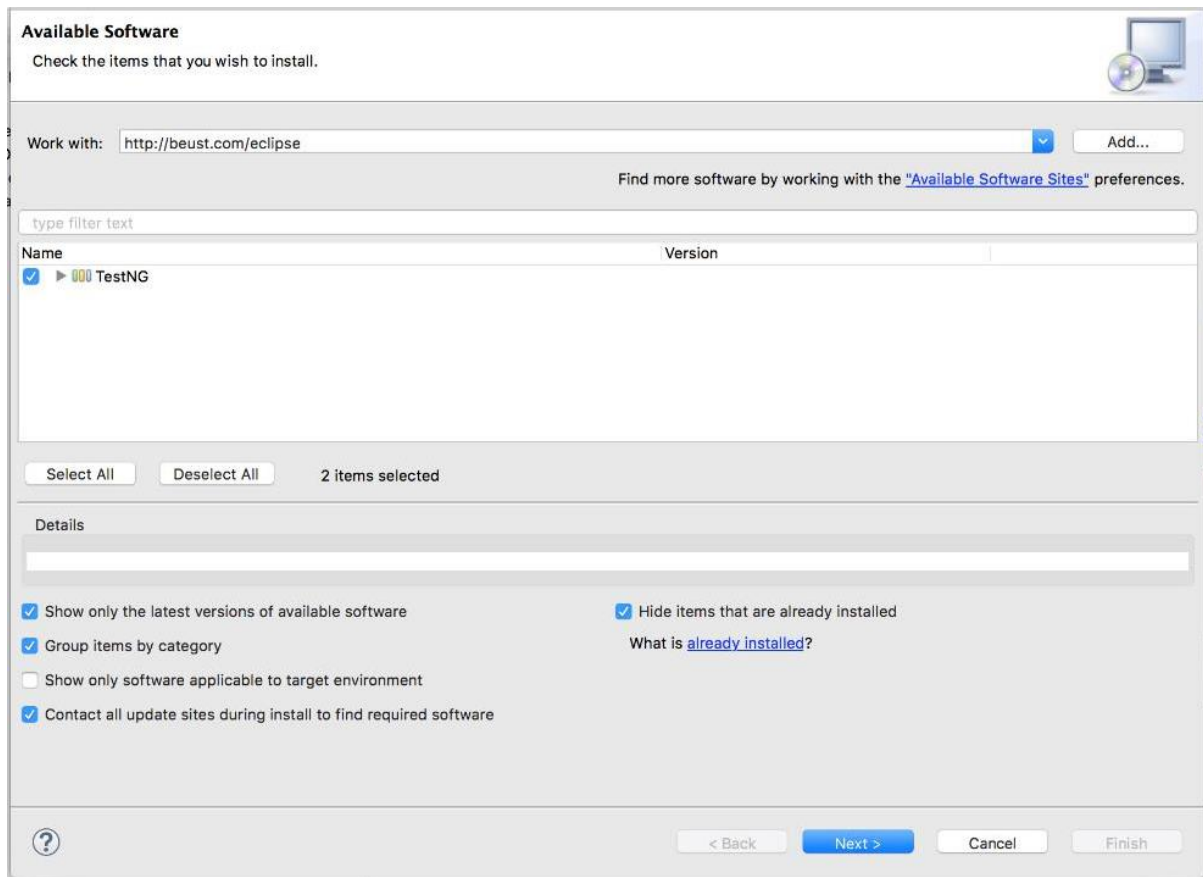
### 3.8 TESTNG PLUGIN CONFIGURATION IN ECLIPSE

It is an open source automated testing framework; where NG of TestNG means Next Generation. TestNG eliminates most of the limitations of the older framework and gives the developer the ability to write more flexible and powerful tests with help of easy annotations, grouping, sequencing & parametrizing. We will use the testNG framework to develop scripts.

Follow the below steps to install testNG plugin in eclipse.

- a. Open Eclipse and Go to Help-> Install New Software-> Enter URL as shown below:
- b. Enter TestNG URL: <http://beust.com/eclipse> in the Work with dropdown as highlighted below.





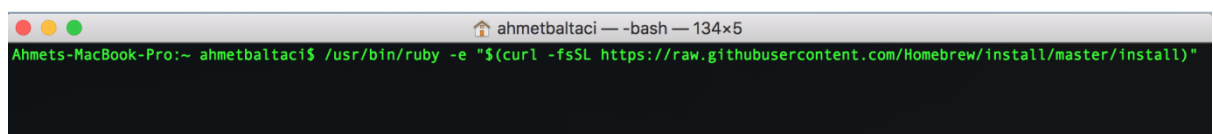
- c. Follow the on-screen instruction and accept the defaults to complete the installation. This should install TestNG plugin to eclipse.
- d. After successful installation, go to Eclipse -> Help -> Eclipse Market place under Installed tab you can see the TestNG installed.

## 3.9 REQUIRED INSTALLATION FOR APPIUM iOS MOBILE AUTOMATION

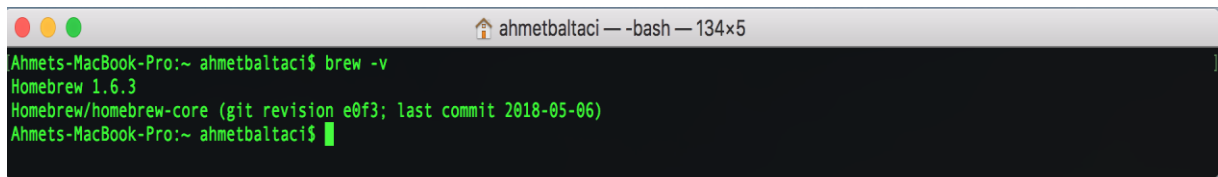
### 3.9.1 Homebrew Installation

- a. In order to install and update the required packages easily, we should install HomeBrew and enable it. Open the terminal and type below command and enter to install HomeBrew.

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```



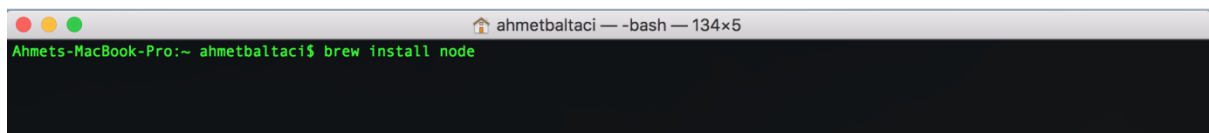
- b. Then, let's type `brew -v` command and hit the enter key. If you will see the version information that means the installation has been done correctly.



```
ahmetbaltaci — -bash — 134x5
Ahmet-MacBook-Pro:~ ahmetbaltaci$ brew -v
Homebrew 1.6.3
Homebrew/homebrew-core (git revision e0f3; last commit 2018-05-06)
Ahmet-MacBook-Pro:~ ahmetbaltaci$
```

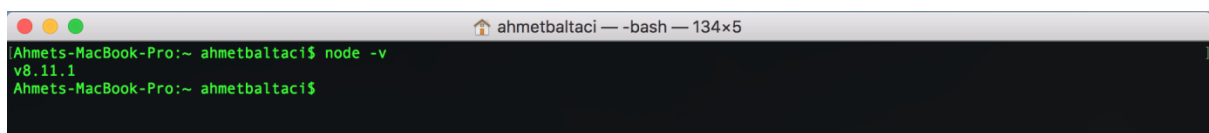
### 3.9.2 Node.js Installation

- a. Option 1: Recommended: We need to write **`brew install node`** command and hit the enter key to install node.js.



```
ahmetbaltaci — -bash — 134x5
Ahmet-MacBook-Pro:~ ahmetbaltaci$ brew install node
```

- b. We can verify the installation by executing `node -v` command on terminal. If we see the version information, that means installation went well and node has been installed.

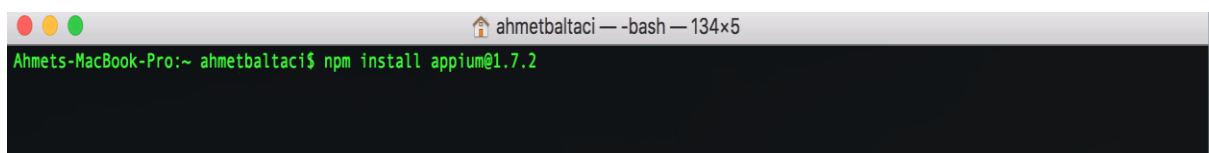


```
ahmetbaltaci — -bash — 134x5
Ahmet-MacBook-Pro:~ ahmetbaltaci$ node -v
v8.11.1
Ahmet-MacBook-Pro:~ ahmetbaltaci$
```

Option 2: Download Node.js from the below URL <https://nodejs.org/en/download/>

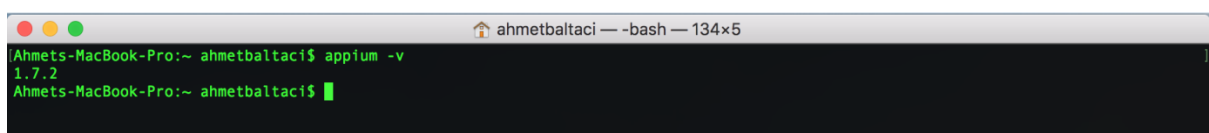
### 3.9.3 Appium Installation

- a. Install specific version of Appium v1.7.2, using this command **`npm install appium@1.7.2`**.  
b. To install latest version of appium type `npm install -g appium@beta`



```
ahmetbaltaci — -bash — 134x5
Ahmet-MacBook-Pro:~ ahmetbaltaci$ npm install appium@1.7.2
```

- c. When the installation finishes, execute **`appium -v`** command to check Appium's installation and its version.



```
ahmetbaltaci — -bash — 134x5
Ahmet-MacBook-Pro:~ ahmetbaltaci$ appium -v
1.7.2
Ahmet-MacBook-Pro:~ ahmetbaltaci$
```

### 3.9.4 Installation and Settings of external dependencies

- a. In order to run our tests on real devices, we need to install libimobiledevice dependency by executing `brew install libimobiledevice --HEAD` command on terminal

`brew install libimobiledevice --HEAD` # install from HEAD to get important updates

```
brew install ideviceinstaller    # only works for ios 9.
```

- b. There is also a dependency, made necessary by Facebook's WebDriverAgent, for the Carthage dependency manager. Install carthage using the below command:

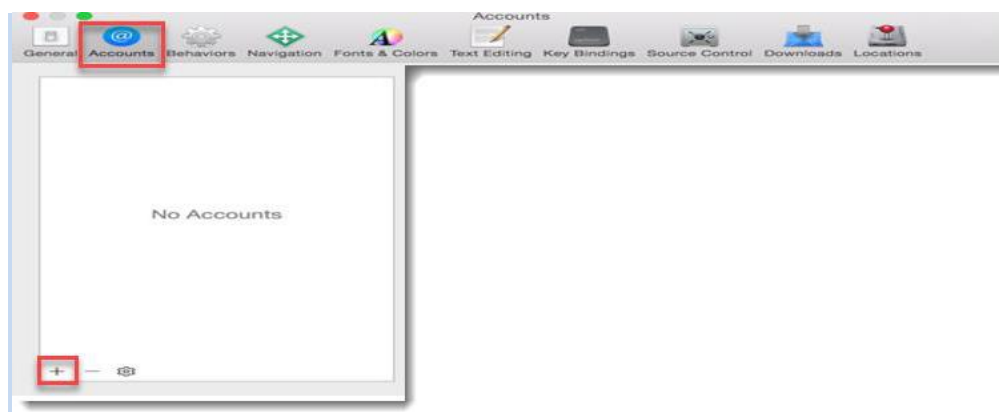
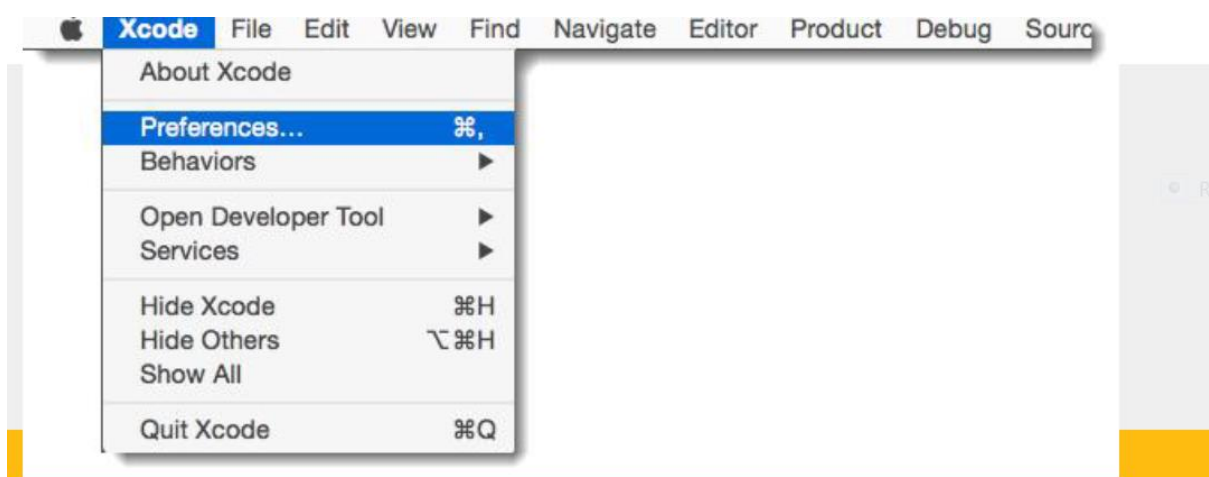
```
brew install carthage  
npm install -g ios-deploy    #required for ios v10.0
```

### 3.9.5 Development certificate and Provisioning profile:

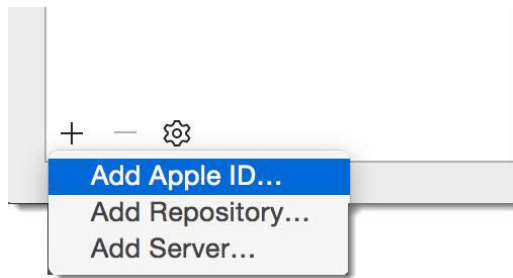
A valid iOS Development Distribution Certificate and Provisioning Profile are necessary to test on a real device. The application will also need to be signed. The developer team provides the code signed IPA of the application.

For building the WebDriver Agent apps, we need iPhone Developer Certificate in the Keychain access. Follow the below steps to get the XCODE generated provisioning profile added to the Keychain Access.

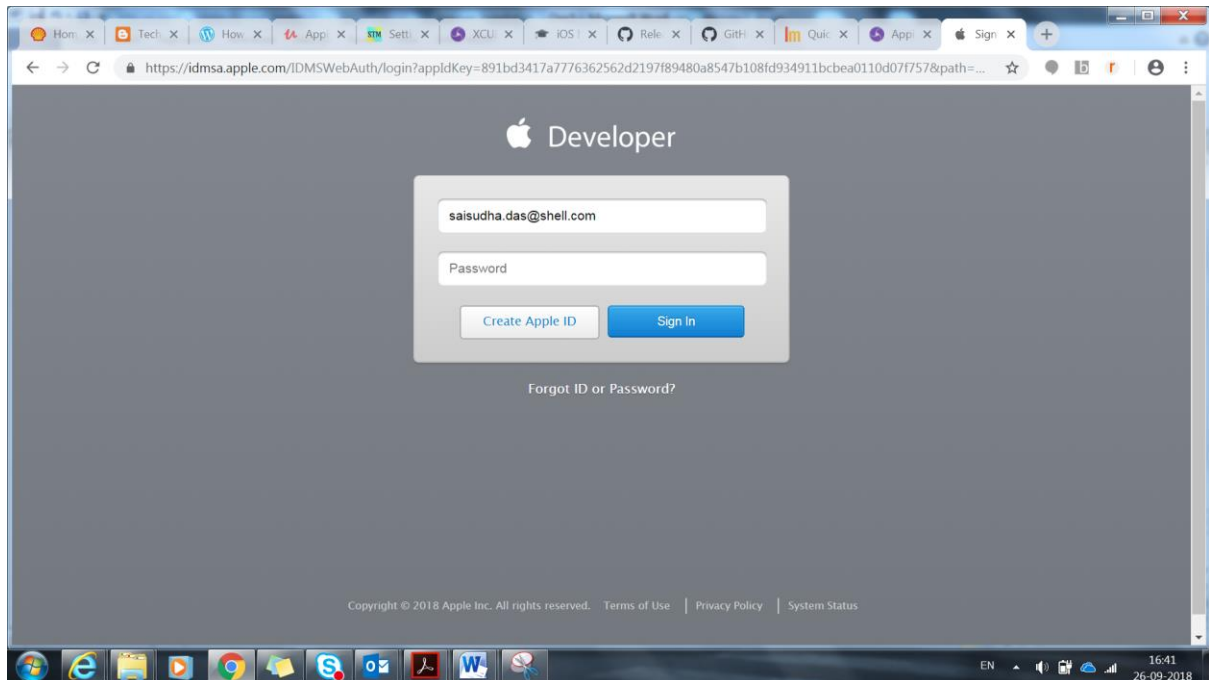
- Add a Developer Account in the xcode.
- Add your developer account by selecting **Preferences...** from the **Xcode** menu and opening the **Accounts** tab



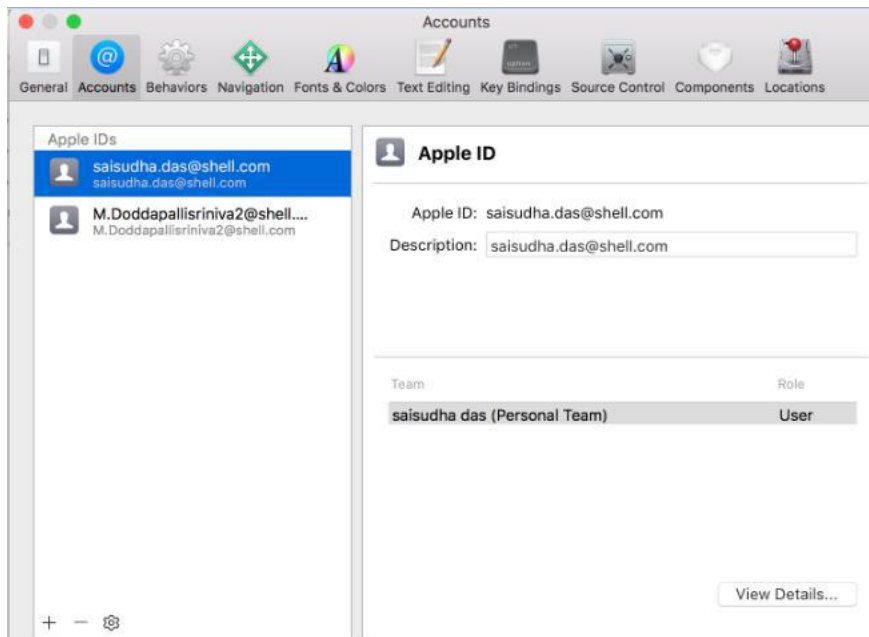
- c. Click on the plus button in the bottom left and choose Add Apple ID... from the list of options. Enter your credentials and click Add.



- d. If you don't have an Apple ID, then go to developer.apple.com ->Accounts Tab- > Create Apple ID.

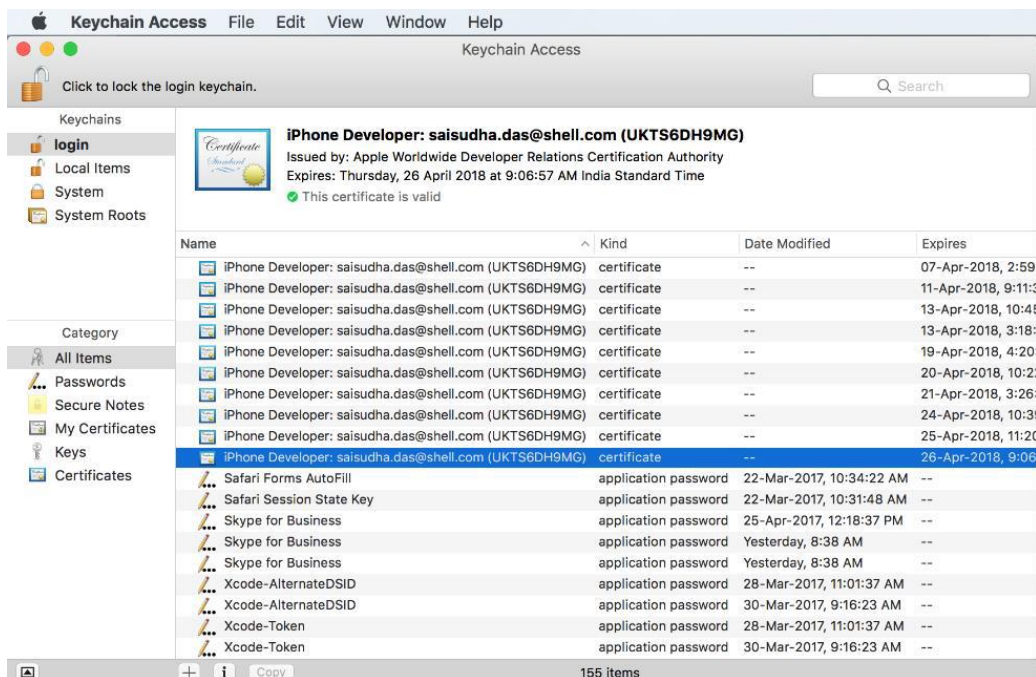


- e. Provide the same Apple ID here and the screen would look like this:



f. Build any XCODE project to Generated Developer Profile in the Keychain access:

- Open the source code of an XCODE project and double click on the “xcodeproj” files.
  - Change the bundle identifier as per the convention of “com.automation.name”. Select the team as the developer profile and the build the application successfully.
- g. Open Keychain Access using Spotlight Search and verify the iPhone Developer Profile is added. The iPhone Developer Certificate should be a valid one as highlighted in the below image:



### 3.9.6 WebDriverAgent

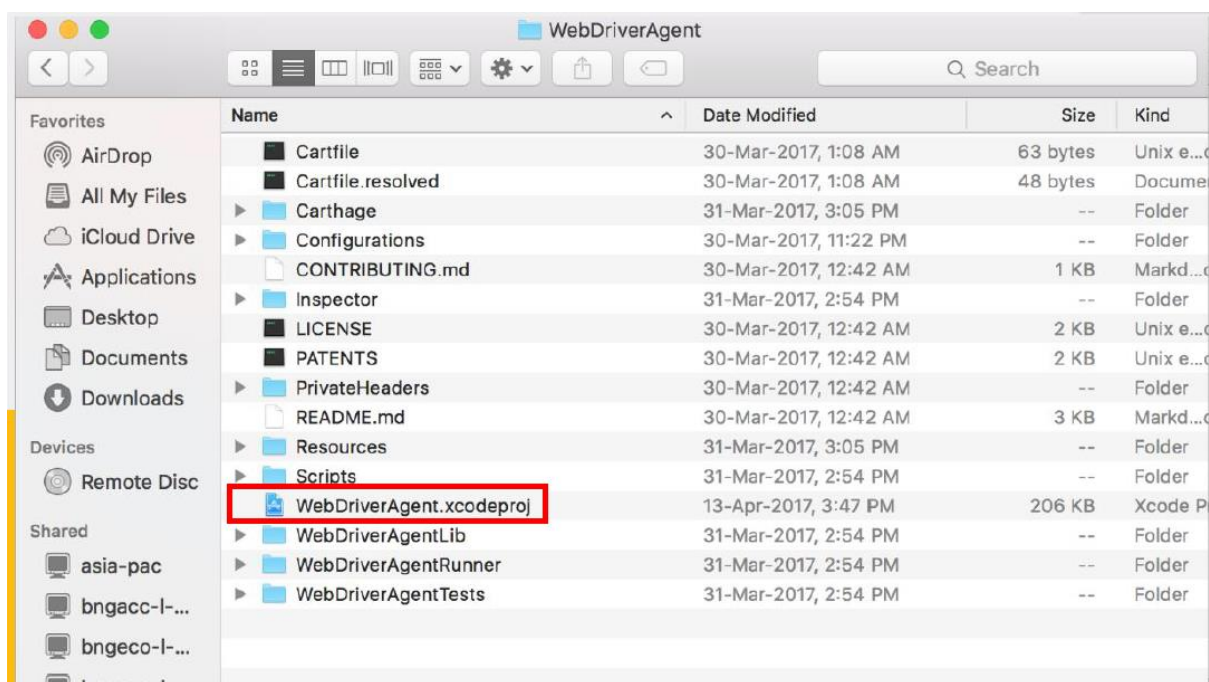
It opens the apps when it is downloaded to the device. It is a WebDriver server application which let us control iOS devices remotely. We can integrate it with XCTest framework and run the commands on the device. WebDriverAgent is developed at Facebook for end-to-end testing. In order to install it, we should do the below steps:

- a. Go to webdriver agent folder using the command: `cd /usr/local/lib/node_modules/appium/node_modules/appium-xcuitest-driver/WebDriverAgent`
- b. Then run the following in order to set the project up:

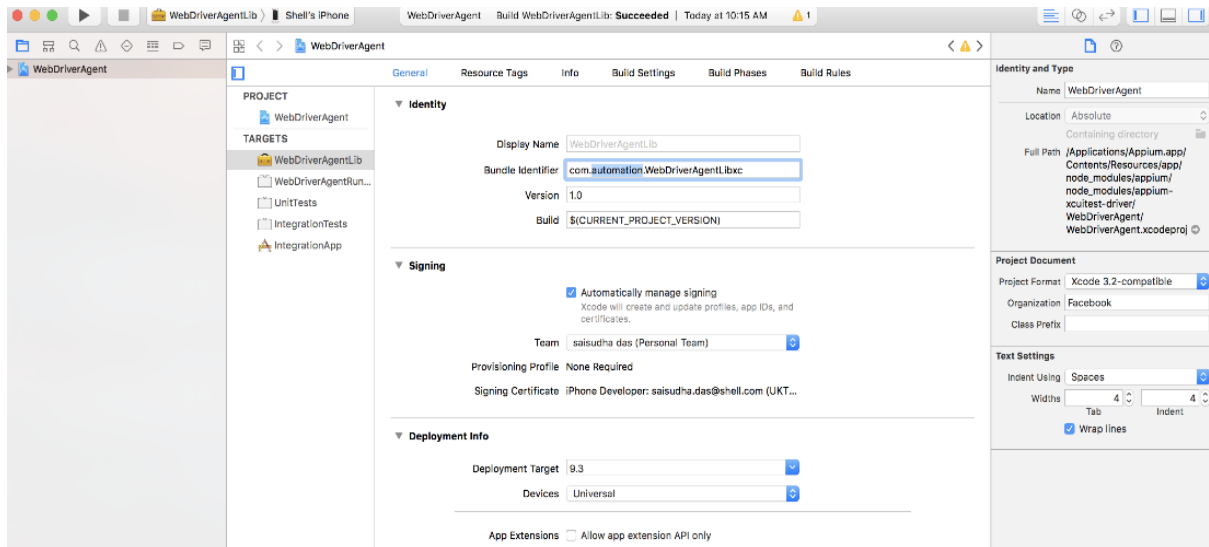
```
mkdir -p Resources/WebDriverAgent.bundle
```

```
./Scripts/bootstrap.sh -d
```

- c. Type `open .` in terminal which will open the WebDriverAgent folder.

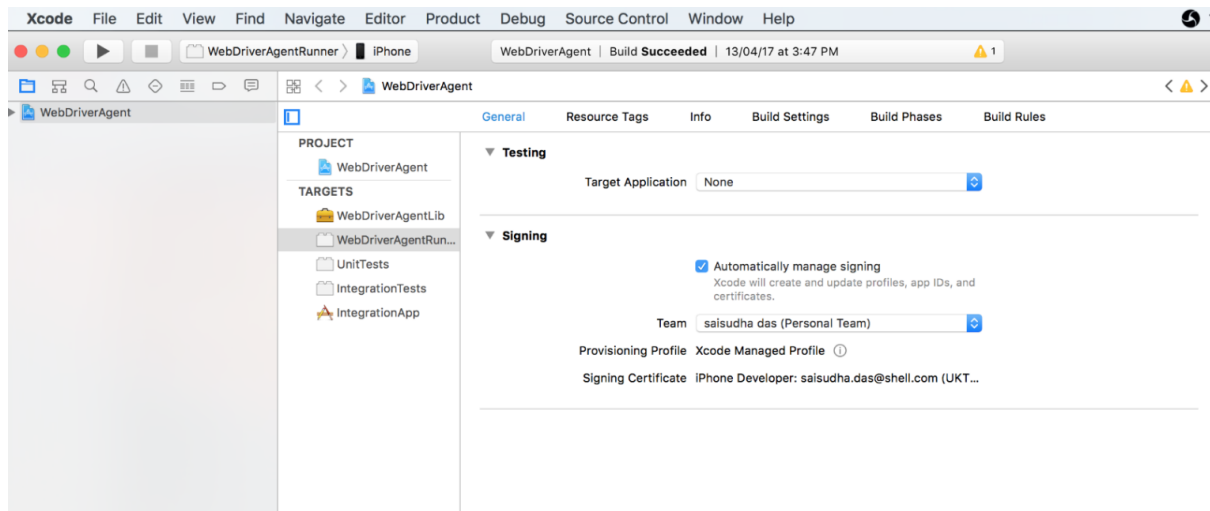


- d. Now double click on `WebDriverAgent.xcodeproj`. This will open the project in xcode



- e. Select WebDriverAgentLib target and update the bundle identifier as "com.automation.WebDriverAgentLibxc" then click on the 'Automatically manage signing' checkbox in the General tab.
- f. Select the Provisioning Profile in the Team drop down
  - iOS devices configurations:  
Follow the below steps to enable UI Automation on the device. Open Settings> Developer on your iOS phone or tablet. Turn on the Enable UI Automation.
- g. Connect your device via USB to your MacBook, select the Device and then build the project by clicking on Play button.
- h. After the successful build of the WebDriverAgentLib, repeat the same steps for WebDriverAgentRunner app and this time update the bundle identifier as "com.automation.WebDriverAgentRunnerxc".

**Note:** If error is encountered during the 'Automatically Manage Signing' step we need change the bundle identifier for both the WebDriverAgentLib and WebDriverAgentRunner apps. For WebDriverAgentLib the bundle identifier can be updated against the "Identity" section under the 'General' tab and for WebDriverAgentRunner the bundle identifier can be changed against the "Product Bundle Identifier" section under the 'Build Settings' tab.



- i. Build WebDriverAgent once to verify all above steps worked:

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

Device UDID can be found in the following xcode path

*Windows->Devices->Identifier*

- j. If this was successful, the output should end with something like:

**Test Suite 'All tests'** started at 2017-01-23 15:49:12.585

**Test Suite 'WebDriverAgentRunner.xctest'** started at 2017-01-23 15:49:12.586

**Test Suite 'UITestingUITests'** started at 2017-01-23 15:49:12.587

**Test Case '-[UITestingUITests testRunner]'** started.

t = 0.00s Start **Test** at 2017-01-23 15:49:12.588

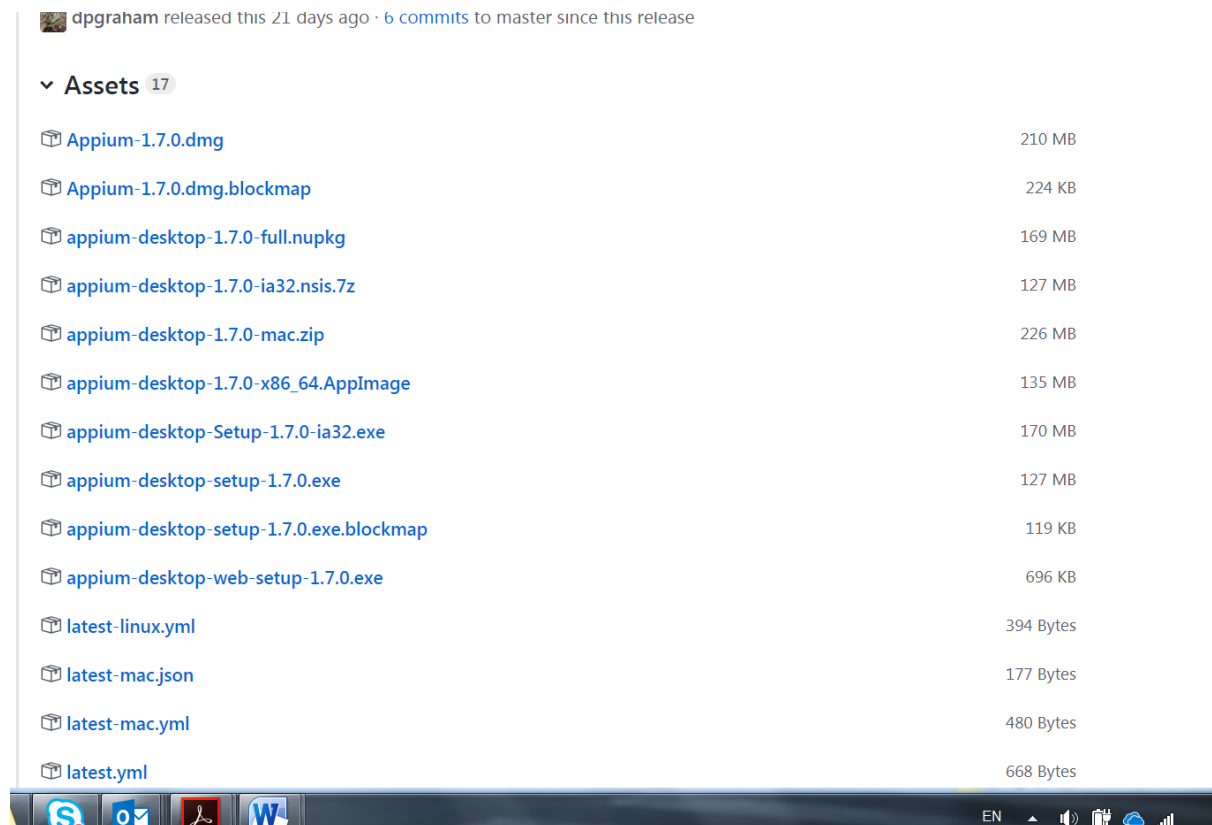
t = 0.00s Set Up

### 3.10 Install Appium Desktop:

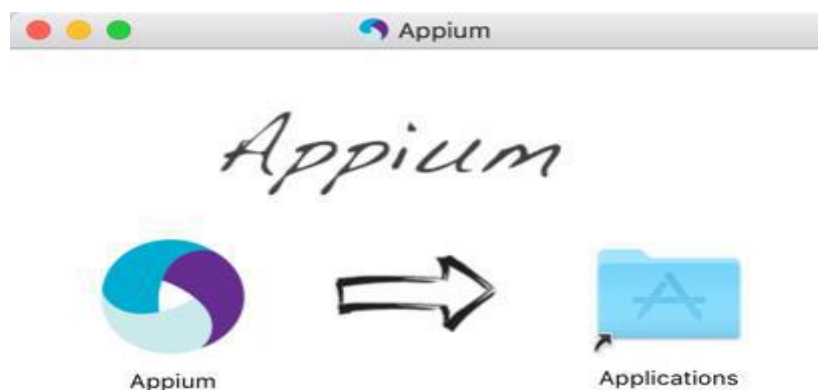
Appium Desktop is an app for Mac, Windows, and Linux which gives you the power of the [Appium](#) automation server in a beautiful and flexible UI. It is a combination of a few Appium-related tools:



- A graphical interface for the Appium Server. You can set options, start/stop the server, see logs, etc... You also don't need to use Node/NPM to install Appium, as the Node runtime comes bundled with Appium Desktop.
  - An Inspector that you can use to look at your app's elements, get basic information about them, and perform basic interactions with them. This is useful as a way to learn about Appium or as a way to learn about your app so you can write tests for it.
- a. Download and install Appium desktop from <https://github.com/appium/appium-desktop/releases>



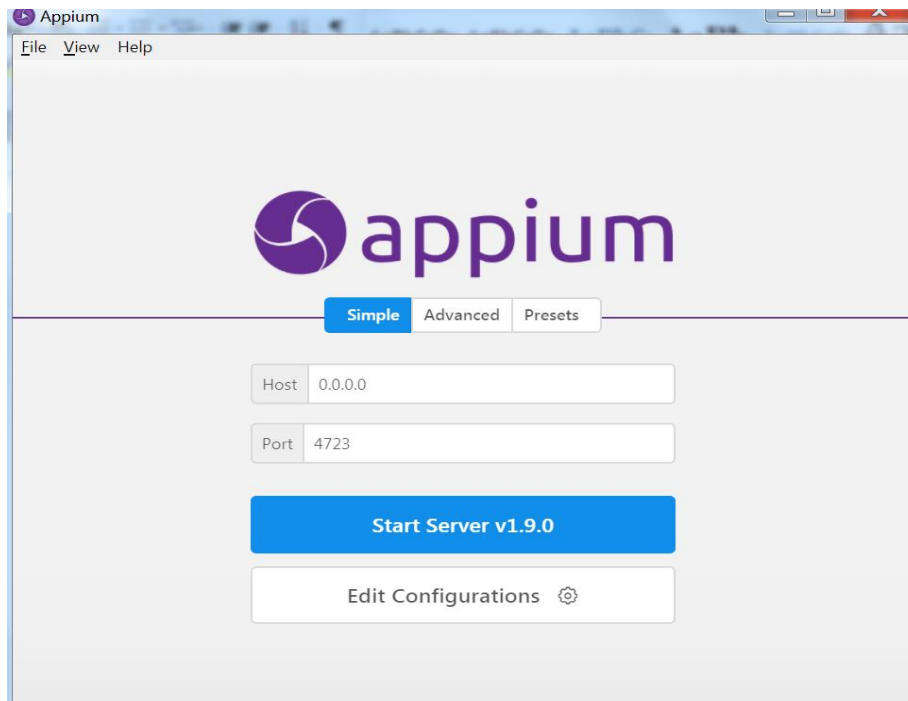
- b. Download and install the dmg file for appium. You will need to install Appium Desktop by copying the app from the downloaded DMG file to your own file system (the best place is the "Applications" folder).



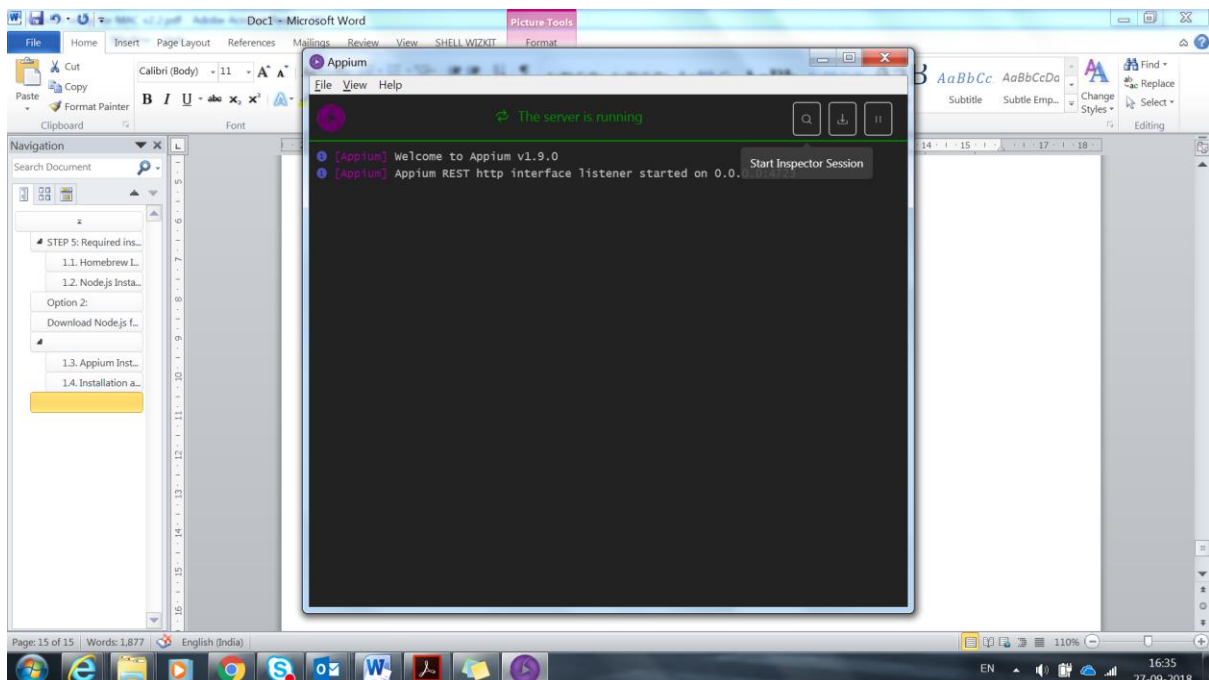
- c. Go to applications and launch appium

### 3.11 Element Inspection for iOS and Android

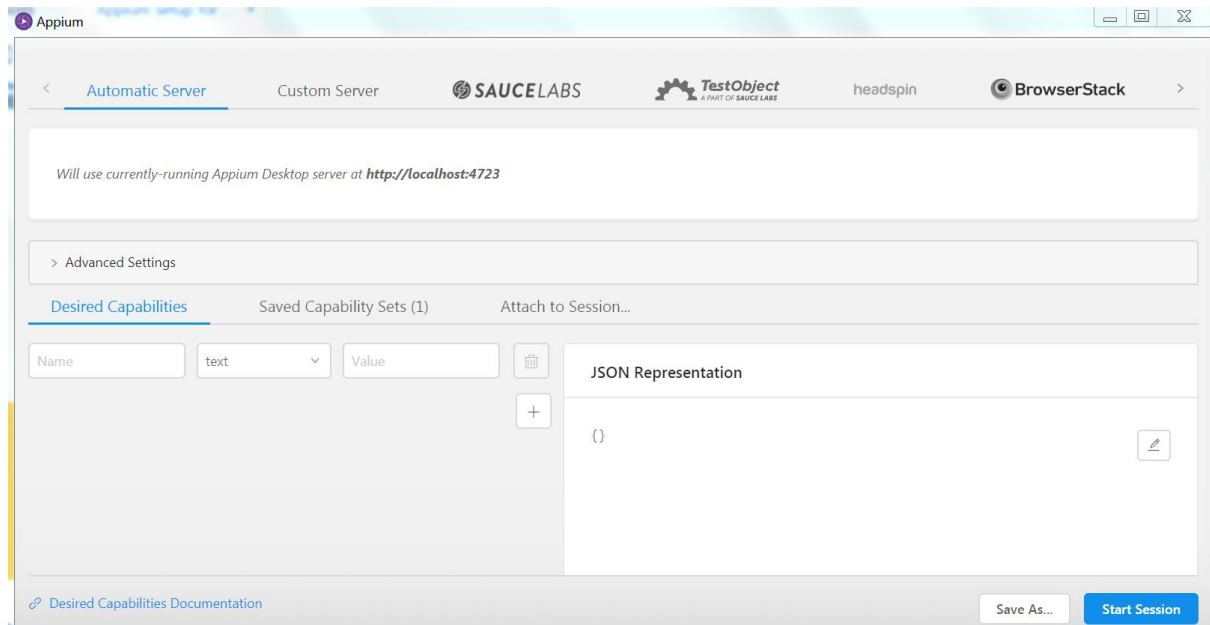
- a. Open Appium Desktop app from the launch pad and click on the 'Start Server v 1.9.0.'



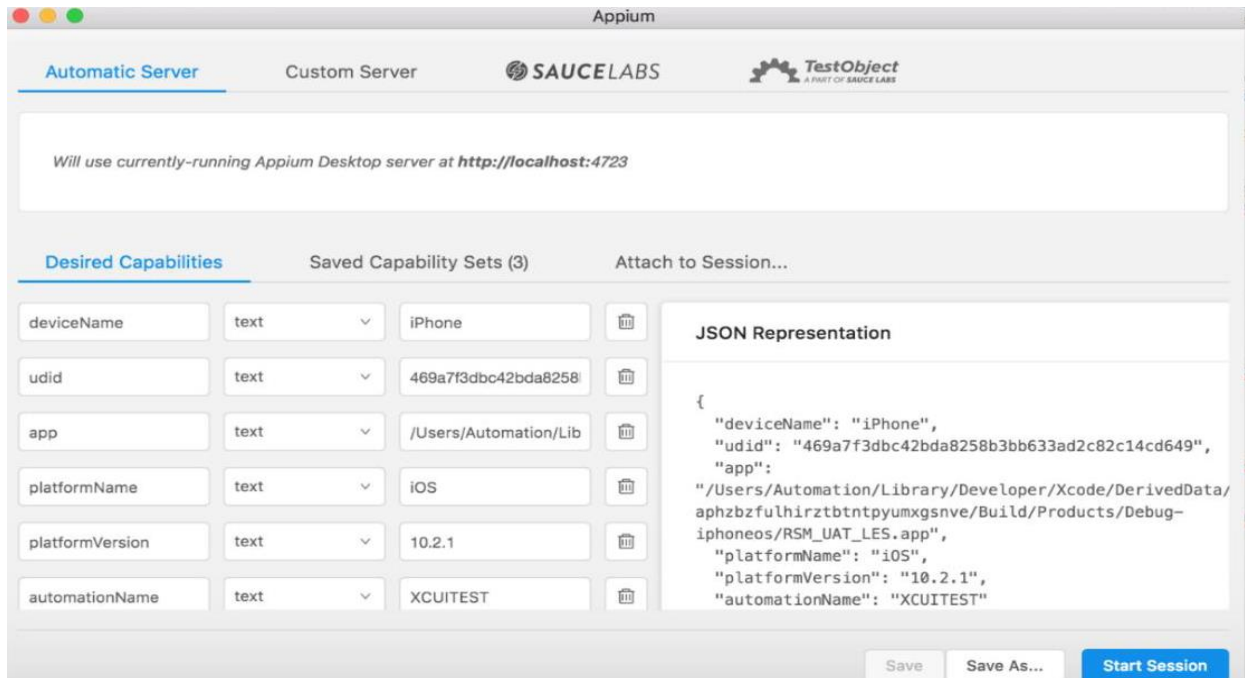
- b. Click on 'Start Inspector Session' button



- c. Set the capabilities for the iOS device in the below screen. To set the capabilities follow the below steps:
- Enter the capability name in the 'Name' text box and the value in the 'value' text box.
  - Click on the '+' icon.
  - Enter all the necessary capabilities and then click 'Save As' button to give a name to the capabilities set for the specific device.



- Make sure the below mentioned capabilities are set properly.
  - deviceName='the name of the device'
  - platformName='iOS'
  - platformVersion='10.2'
  - udid='udid of the device'
  - app='path of the code signed ipa file' OR 'path of the app file generated'
  - automationName='XCUITEST'
- For Android: Make sure the below mentioned capabilities are set properly.
  - deviceName='the name of the device'
  - platformName='Android'
  - platformVersion='7.1'
  - app='path of the apk file'
  - automationName=uiautomator2



d. Click on the 'Start Session' button. And start element inspection of the app under test.

