Automation Setup Documentation

This Document helps users with the steps to run the Molekule mobile app Automation script from scratch for both MacBook and Windows host machines.

# Mac OS(iOS):

## Prerequisites:

### Install Xcode

* + - Open App Store
    - Type xcode in the search box and press enter
    - Select Xcode and click on Get / Install
    - Open terminal and execute below command
      * **$ xcode-select --install**

### Install Python

* + - Install Python3 on Mac referring following steps:

1. [**Install python on Mac**](https://installpython3.com/mac/)
2. Check whether the Python is already installed in the system or not by typing the command “python” in the terminal.
3. Also check version of python using the command “--version” as shown below

## Steps to Setup Appium on Mac:

## Install brew on MacBook copying and running below command on terminal:

* + - $ /bin/bash -c "$(curl –fsSL <https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh>)"

## Install Node:

* + - $ brew install node
    - $ node –v

## Install Appium using node

* + - $ npm install -g appium
    - $ appium -v

## Install appium doctor

* + - $ npm install -g appium-doctor

## Install Carthage

* + - $ brew install carthage

## Set up Allure framework

* + - $ brew install allure

XCUITest Driver Real Device Setup

* Installing dependencies
* Creating webdriver agent on real device

## Installing dependencies

* + - Appium iOS real device support depends on a central third-party software suite,libimobiledevice , which is easily installable with Homebrew :

$ brew install libimobiledevice

* + - In addition to the dependency on libimobiledevice, Appium support for real devices running iOS 9.3 and above using Xcode 8+ also depends on ios-deploy, which is easily available through npm or Homebrew:

$ brew install ios-deploy

Creating webdriver agent on real device

* + - Appium will not interact directly with iOS real device application, to do so WebDriver agent app has to be created which will provide the interface between appium and applications.
    - Following are the steps to create WebDriverAgent on a real device.
      * Given this installation location, /path\_where\_installed/appium, WebDriverAgent will be found in go to that location and following are steps to follow
        + Find out where your Appium installation is:

Type the command on terminal: $ which appium This command will give the location of where appium is installed on your system.

Type the following command in terminal to find location of all the modules of appium: $ locate appium

Change the current directory to appium node module. Example: Type the following command in terminal:$ cd /usr/local/lib/node\_modules/appium/node\_modules/

Type the following command on terminal: $ ls This command will list all the directories in side /appium/node\_modules/ folder

Change the directory to (WebDriverAgent parent directory) appium-xcuitest-driver or appium-webdriveragent by Typing the following command in the terminal: cd appium-xcuitest-driver or cd appium-webdriveragent

List out the directories inside the directory appium-xcuitest-driver or appium-webdriveragent by typing the command ls in the terminal.

Follows change the directory to WebDriverAgent by typing the following command in the terminal: cd WebDriverAgent/

* After changing directory to WebDriverAgent , execute the following commands in order to set the project up:

i) $ mkdir -p Resources/WebDriverAgent.bundle

$ ./Scripts/bootstrap.sh -d

* Open WebDriverAgent.xcodeproj in Xcode.

Type open . command in the terminal to find the location of WebDriverAgent.xcodeproj.

Open WebDriverAgent.xcodeproj in xcode.

WebDriverAgent WebDriverAgent.xcodeproj Right click on WebDriverAgent.xcodeproj [ Open with Xcode ] 

This necessitates manually changing the bundle id for the target from the "Build Settings" tab, and changing the "Product Bundle Identifier" from com.facebook.WebDriverAgentRunner to something that Xcode will accept. For example, a Bundle Identifier is like - “io.appium.WebDriverAgentLib” .

For both the WebDriverAgentLib and WebDriverAgentRunner targets, select "signing(Debug) and Signing(Release)" in the "General" tab.

a. Now click on WebDriverAgentLib.In WebDriverAgentLib > signing(Debug) select the Team (Provide the team name) and also select Signing Certificate as iOS developer.

b. And also In WebDriverAgentLib > signing(Release) select the Team (Provide the team name) and also select Signing Certificate as iOS developer

c. Now click on WebDriverAgentRunner.In WebDriverAgentRunner>In signing(Debug)> select the Provisioning Profile (Provisional certificate should be created before).

d. And also in WebDriverAgentRunner > signing(Release)> select the Provisioning Profile.

Now click on the build icon of xcode. It will show build succeeded toast message.

Finally, build the project by entering the following command on the terminal.

$ xcodebuild -project WebDriverAgent.xcodeproj -scheme WebDriverAgentRunner -destination 'id=udid' test

In this above command udid has to be replaced with the connected device udid.

If it is successful

To verify, try accessing the WebDriverAgent server status.

a. export DEVICE\_URL='http://:8100' Replace 'http://:8100' with ServerURLHere i.e 'http://192.168.1.44:8100' as shown in the above figure.

b. Type the below command in terminal export JSON\_HEADER='-H "Content-Type:application/json;charset=UTF-8, accept:application/json"'

c. Type the below command in the terminal and the success screen will be displayed as shown below. curl -X GET $JSON\_HEADER $DEVICE\_URL/status

Finally, the WebDriverAgent app should successfully be created on your mobile.

# Windows OS(Android):

# Install JDK (Java Development Kit)

* + - Go to [link](https://www.oracle.com/java/technologies/javase-downloads.html). Click on JDK Download for Java JDK download.
    - Once the Java JDK download is complete, run the exe to install JDK. Click Next
    - Select the PATH to install Java in Windows… You can leave it Default. Click next
    - Once you install Java in windows, click Close
    - How to set Environment Variables in Java: Path and Classpath
    - Check whether java is installed on the system or not by typing java on the terminal.
      * $ java --version

# Install ANDROID SDK (Studio)

* + - Go to the [link](https://developer.android.com/studio). Click on DOWNLOAD ANDROID STUDIO and install.
    - Setup Path for Android SDK (ANDROID\_HOME, Platform-tools, and Tools)
    - Check whether android is installed on the system or not by typing $ adb on the command prompt.

# Install Python

* + - Install Python3 on Mac referring following steps:
      * [**Install python on Windows**](https://installpython3.com/windows/)
      * Check whether the Python is already installed in the system or not by typing the command “python” in the terminal.
      * Also check version of python using the command “--version” as shown below

# Install Appium

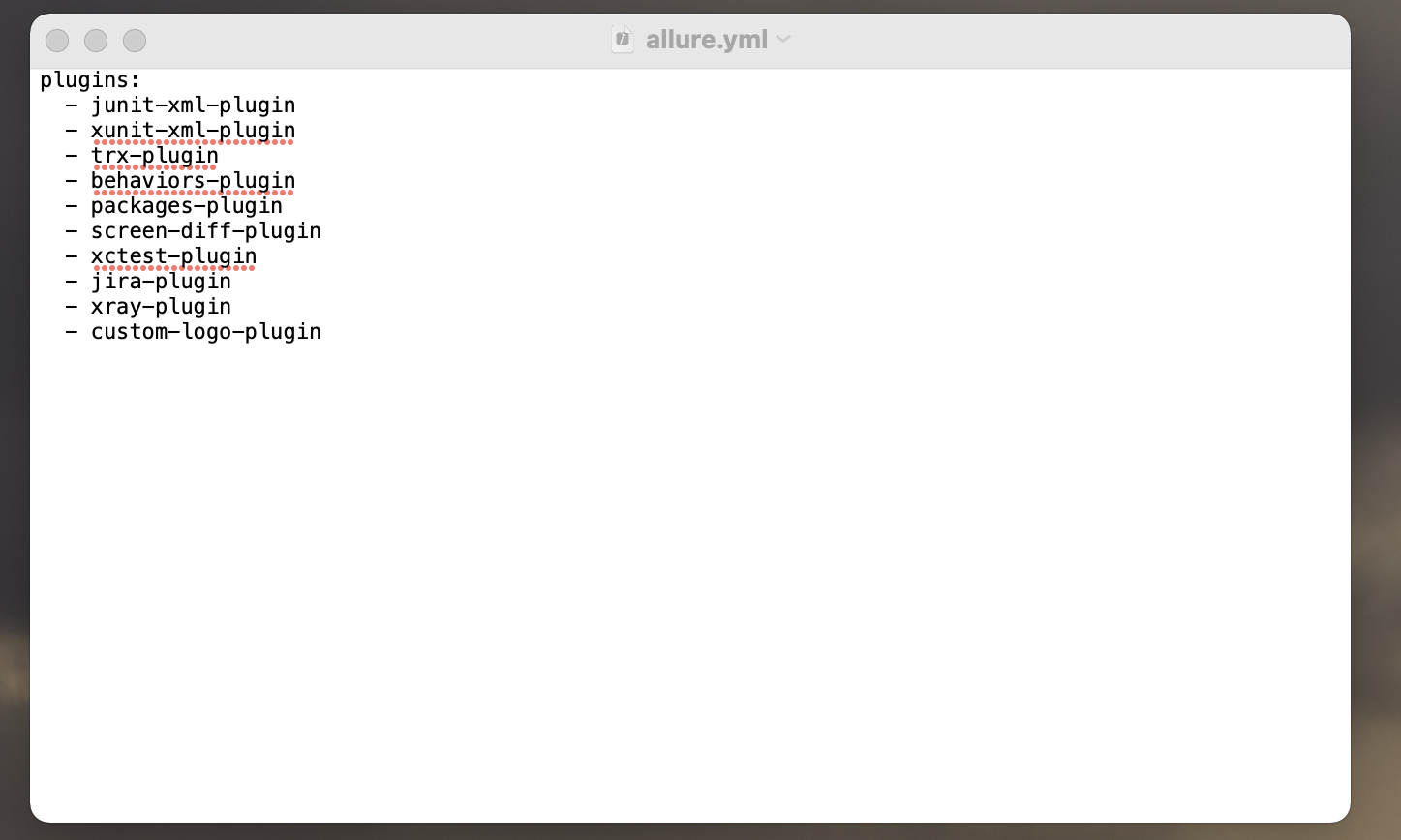
* + - Download node.js from [link](https://nodejs.org/en/download/) for windows
    - Run the installer and install node & npm.
    - Once the installation is completed, open terminal and type
      * $ node –version and $ npm –version
    - Install appium using following command:
      * $ npm install –g appium
    - Check appium if appium is installed or not by using command:
      * $ appium –v

# Install Cygwin

* + - Download and install [Cygwin](http://www.mcclean-cooper.com/valentino/cygwin_install/) for windrows
    - Add location of the Cygwin\bin folder path to the environment PATH variable.

# Set up Allure framework

* + - Download and install allure from [link](https://repo.maven.apache.org/maven2/io/qameta/allure/allure-commandline/2.13.5/)
    - Add location of the allure\bin folder path to the environment PATH variable.
    - Replace the custom-logo-plugin folder in allure directory with custom-logo-plugin in tools directory of automation code
    - Open allure\config\allure.yaml file and add custom-logo-plugin like in the below screenshot



# Common for both Mac and Windows:

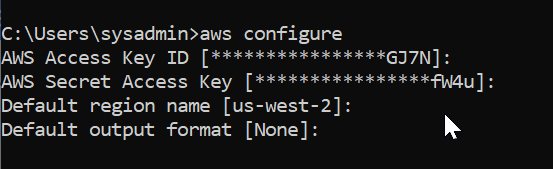
# Set up the aws cli

* + - Download and install the [AWS cli](https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-windows.html)  for windows/mac
    - To verify AWS cli installed or not on your host machine with below command:

$ aws —version

* + - To configure the AWS cli run below command and configure secret key, access key and region:

$ aws configure



# Install Pycharm

* + - Pycharm is an IDE tool kit mainly for developing programs in python.
    - Python script has to be written to launch the application using the pycharm.
    - Open Link - <https://www.jetbrains.com/pycharm/download/>
    - Download the community edition of Pycharm for Mac/Windows operating system
    - Open the downloaded file
    - Go to the Applications directory and double click on “PyCharm”.

# Install the following python packages in the pycharm.

* + - Open Pycharm Preferences Project Python Interpreter Install (+)

|  |  |  |
| --- | --- | --- |
| Appium-Python-Client | 1.1.0 | 1.1.0 |
| PyYAML | 5.4.1 | 5.4.1 |
| allure-python-commons | 2.8.40 | 2.8.40 |
| allure-robotframework | 2.8.40 | 2.8.40 |
| Attrs | 20.3.0 | 20.3.0 |
| Decorator | 5.0.7 | 5.0.7 |
| Docutils | 0.17.1 | 0.17.1 |
| et-xmlfile | 1.1.0 | 1.1.0 |
| importlib-metadata | 4.0.1 | 4.0.1 |
| Natsort | 7.1.1 | 7.1.1 |
| Openpyxl | 3.0.7 | 3.0.7 |
| Pip | 19.0.3 | 21.1.1 |
| Robot | 20071211 | 20071211 |
| Robotframework | 4.0.1 | 4.0.1 |
| robotframework-appiumlibrary | 1.5.0.7 | 1.5.0.7 |
| robotframework-excellibrary-xwfintech | 0.0.3 | 0.0.3 |
| robotframework-python3 | 2.9 | 2.9 |
| robotframework-pythonlibcore | 2.2.1 | 2.2.1 |
| robotframework-selenium2library | 3.0.0 | 3.0.0 |
| robotframework-seleniumlibrary | 5.1.3 | 5.1.3 |
| Selenium | 3.141.0 | 3.141.0 |
| Setuptools | 40.8.0 | 56.0.0 |
| urllib3 | 1.26.4 | 1.26.4 |
| Xlrd | 2.0.1 | 2.0.1 |
| Xlutils | 2.0.0 | 2.0.0 |
| Xlwt | 1.3.0 | 1.3.0 |

* + - If you run the setup.sh file in the automation folder, that will install all above packages.

# Running automation scripts

* + - Change the values for platform, purifier platformVersion, deviceName and country in the automation/inputs/manifest.xml
    - Start the Appium Server: $ ./start\_appium.sh
    - Change the directory to framework\_utils:
    - $ cd automation/testsuites/libraries/framework\_utils
    - Choose the test\_suites and tags/test cases to run by changing the content of select\_testcases\_to\_run.json file.
    - Execute ruuner.py file to start running test\_suites: $ python3 runner.py