# Title: EXECUTION OF ESPRESSO TEST CASES IN CLOUD

**Why cloud and Why not emulator**

Real devices are good for testing your application under less than optimal network conditions, emulators can’t fully replicate device hardware. This makes it difficult to test against real-world scenarios using an emulator. Issues related to the kernel code, the amount of memory on a device, the Wi-Fi chip, and other device-specific features can’t be replicated on an emulator. It’s not enough to test on emulators alone, Providers Like Google firebase Test labs are providing a variety of real and virtual devices in Free plan, Will have chance to see how Espresso test cases behave with different android mobile manufactures

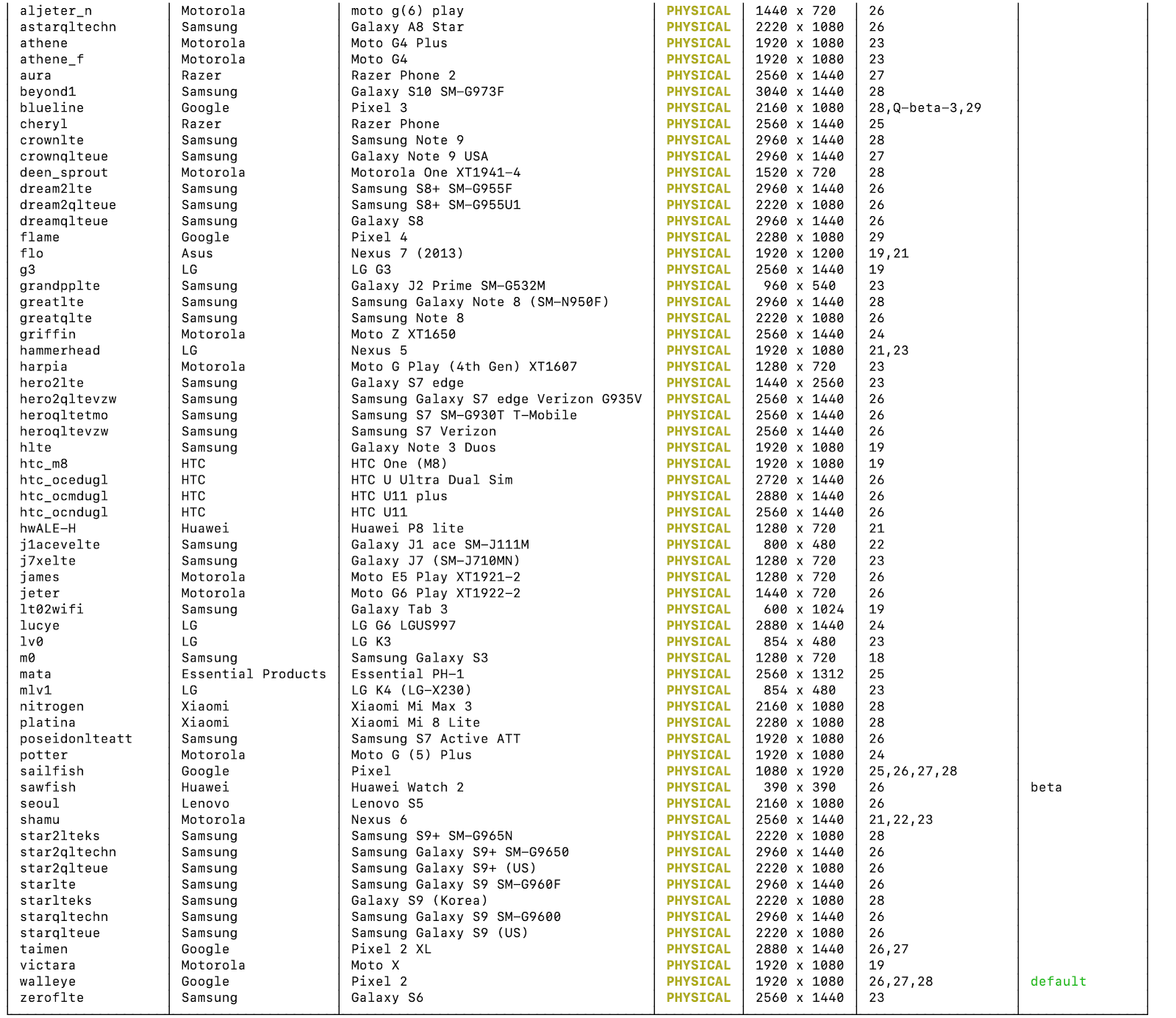
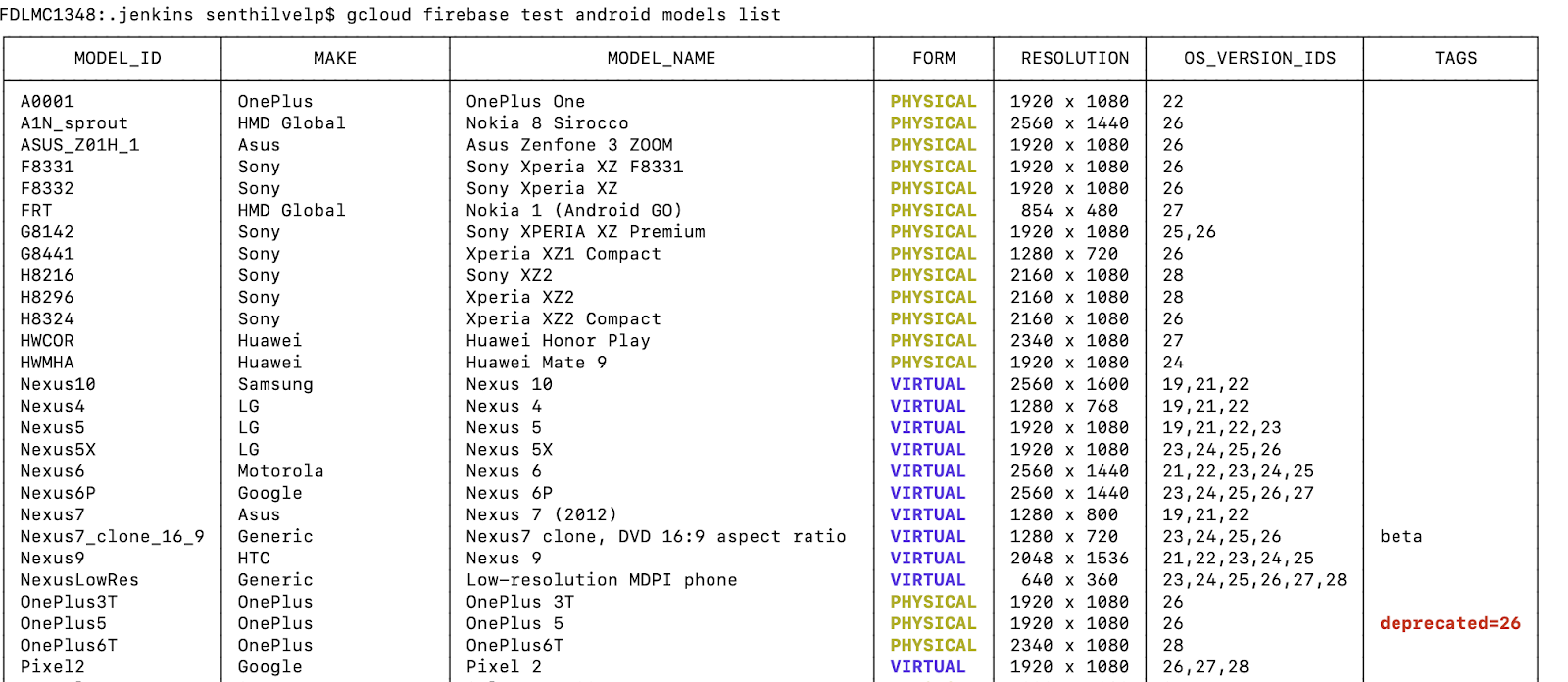
Cloud provides good traceability of automation failures and Execution of Reports than implementing custom reporters in Local emulators, Running Test cases in Parallel Process are provided in the cloud platform

Freshsales has done cloud execution in most of the available cloud platforms like

1. Google firebase Testlabs
2. Browserstack
3. AWS Device farm
4. Sauce labs

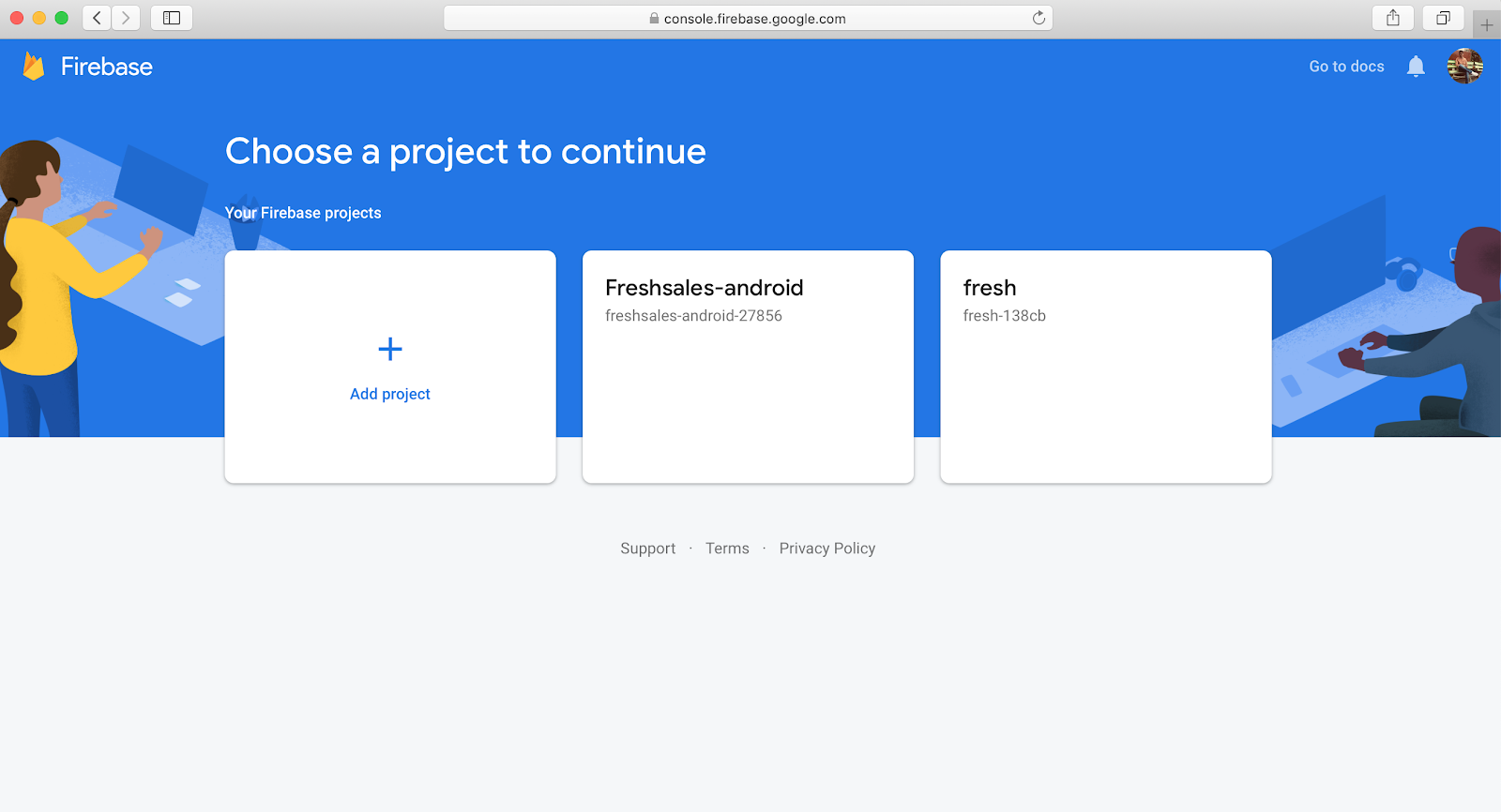
Will discuss each platform in detail regards to their execution style, number of devices available in the free plan and the price of the paid plans and ease of use

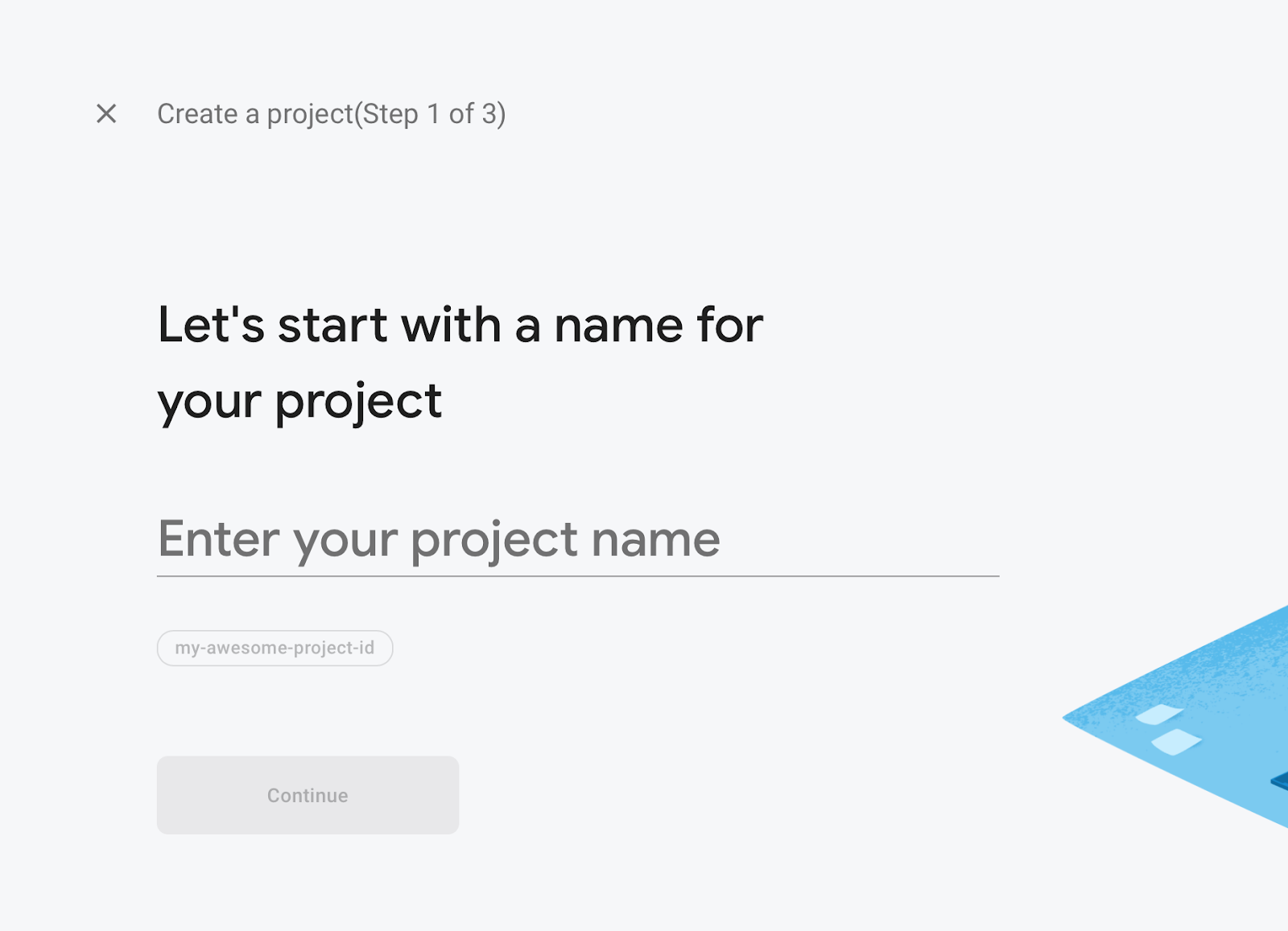
**List of devices provided by Google firebase TestLabs in their free plan**



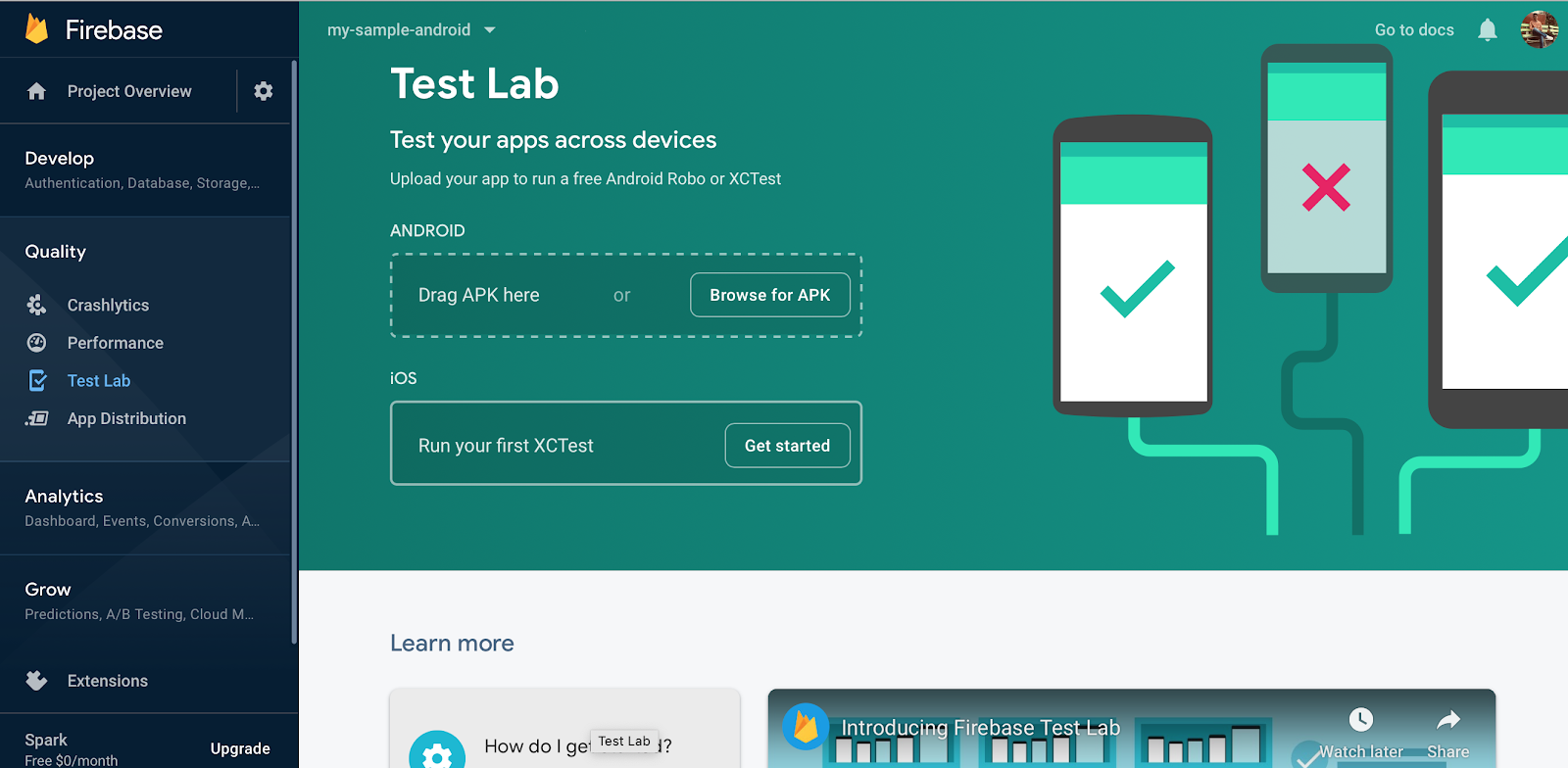
### GOOGLE FIREBASE TESTLABS

Navigate to [https://console.firebase.google.com/](https://www.google.com/url?q=https://console.firebase.google.com/&sa=D&ust=1575459747510000) and create a project directory





After creating a project , tap on the Test lab in the navigation panel

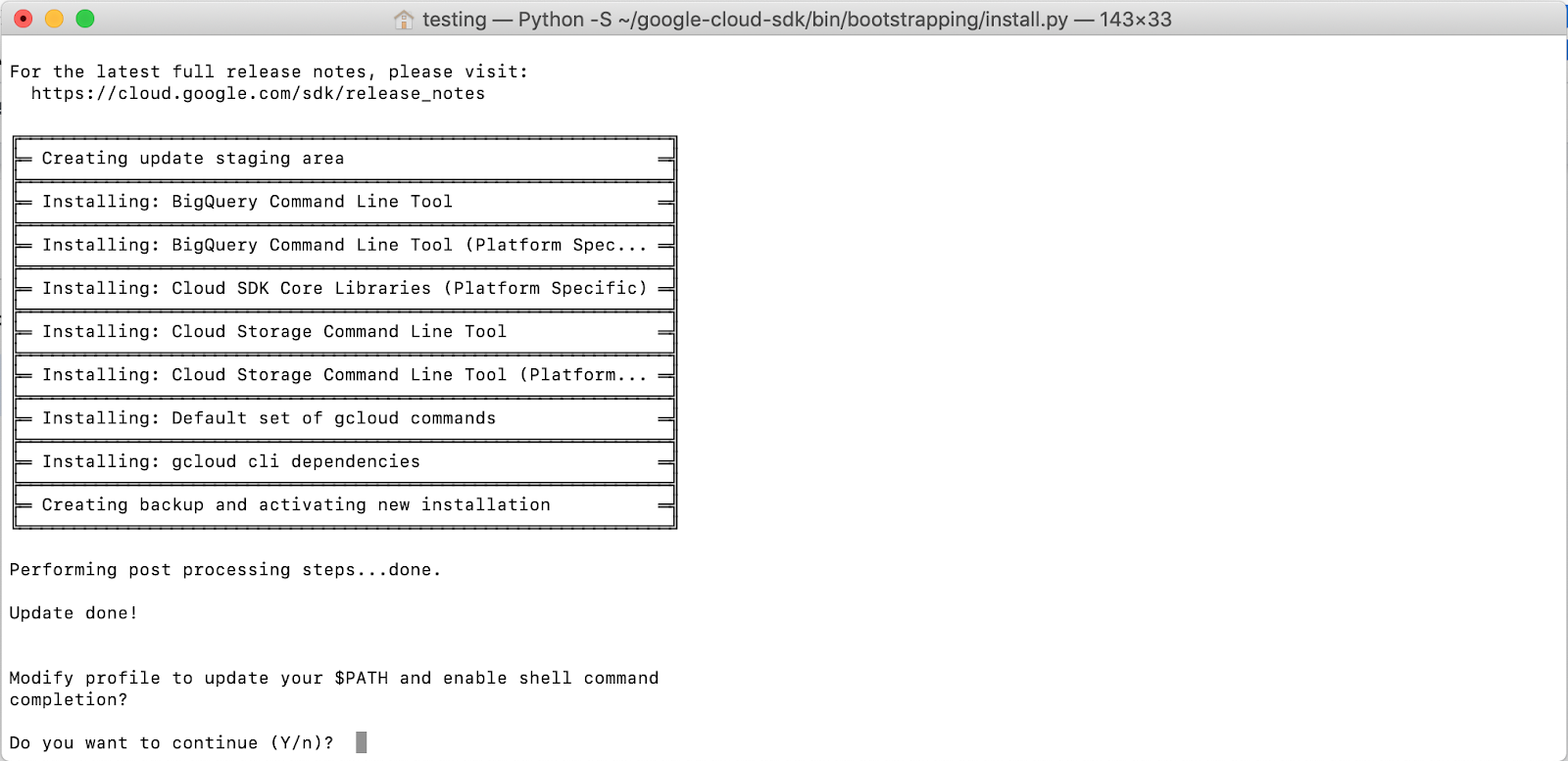


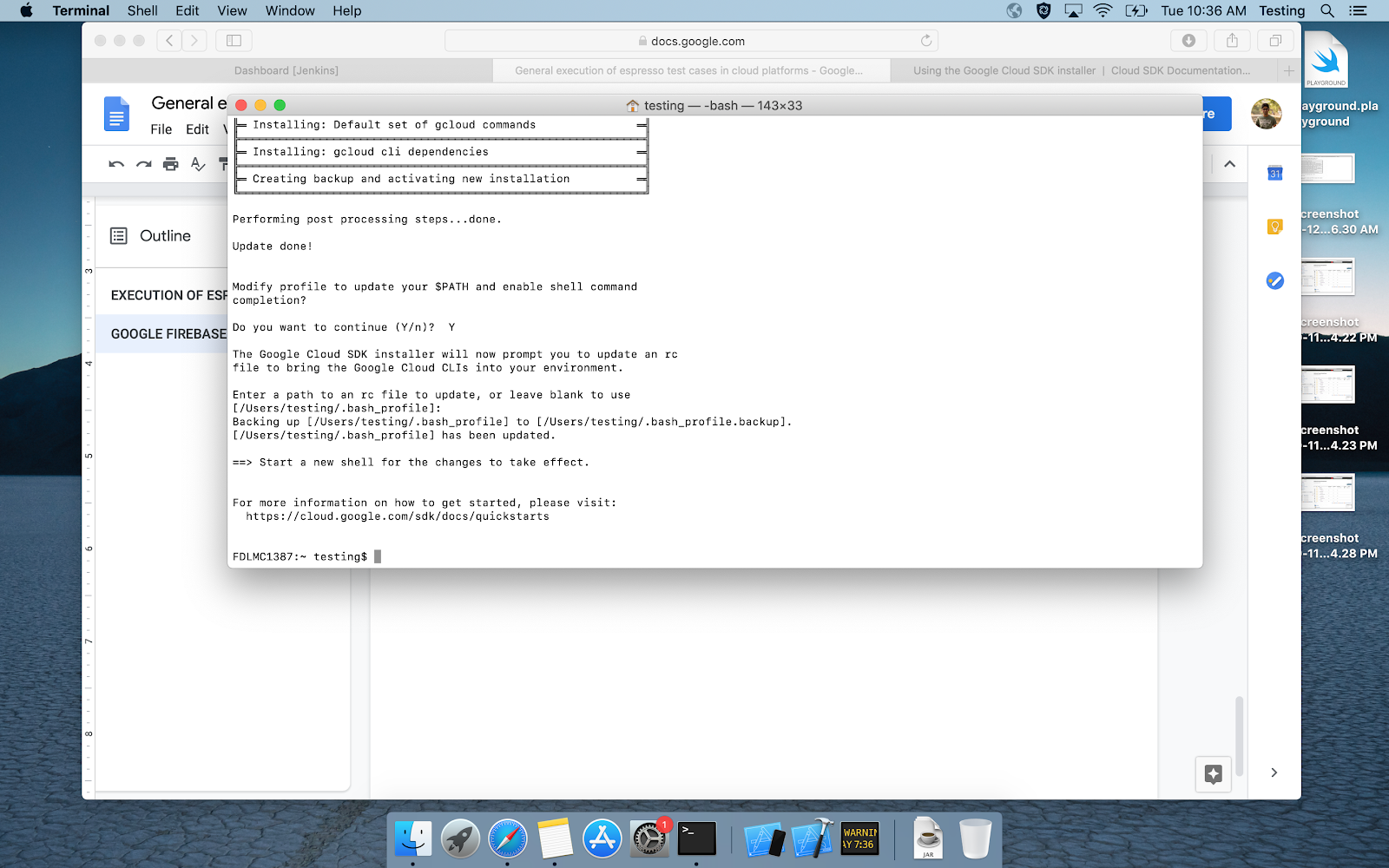
Google firebase allows particular Test class or Test method to be executed via gcloud sdk , So we have to install gcloud sdk in the machine

Type in the terminal

curl <https://sdk.cloud.google.com> | bash

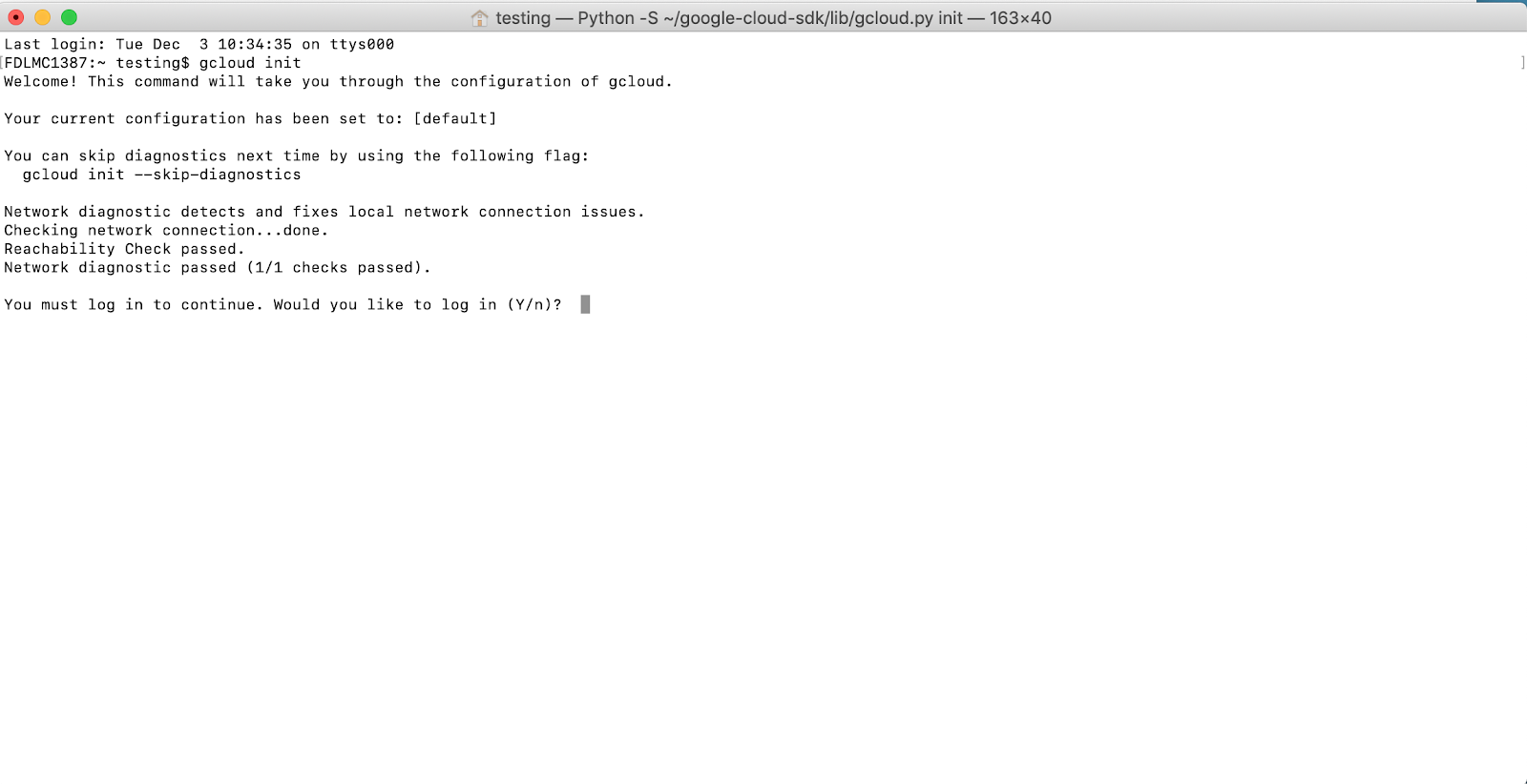
In terminal while installing gcloud sdk with the above command , it would ask to update the $PATH for gcloud command to be used in the terminal , So give “Y”

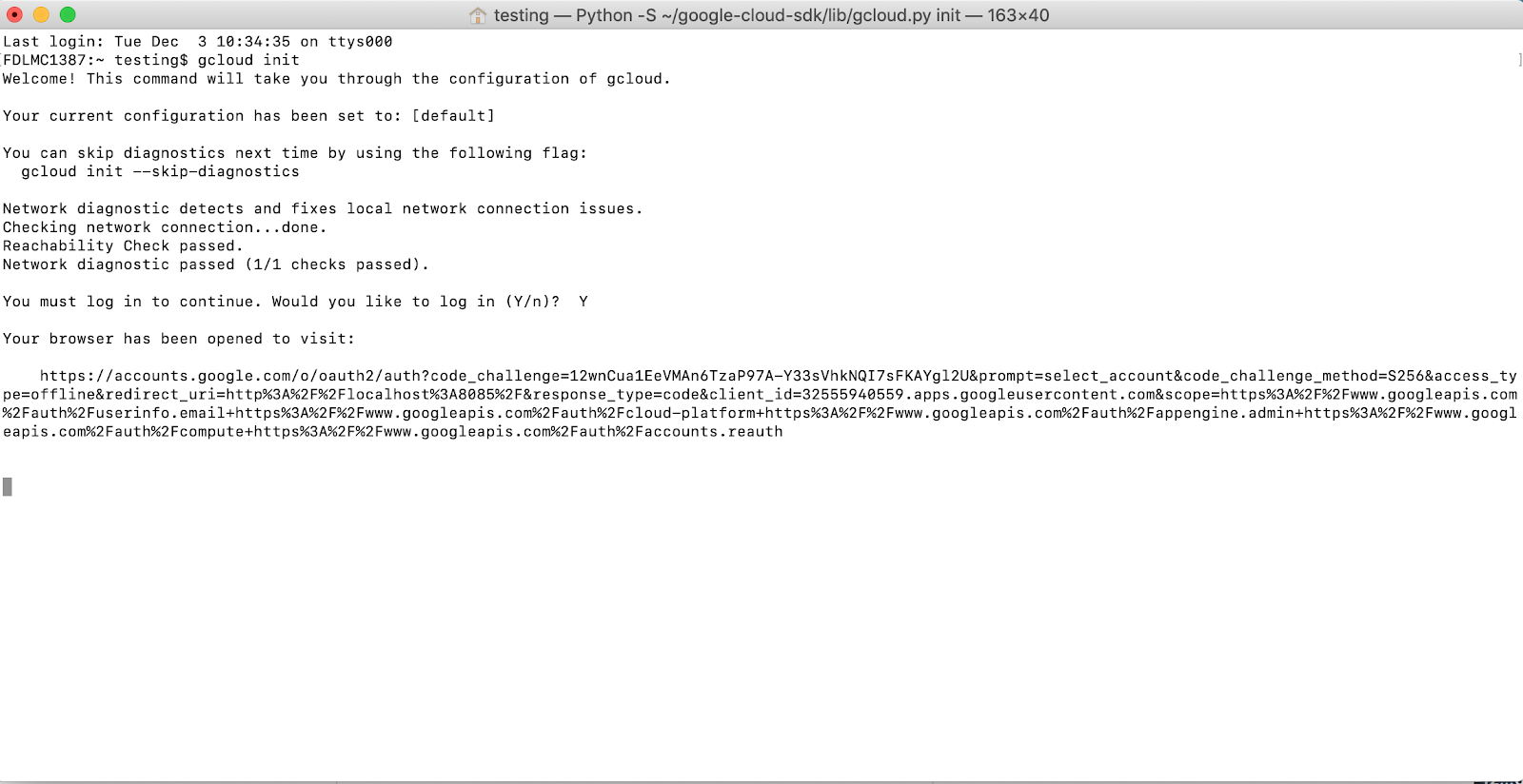


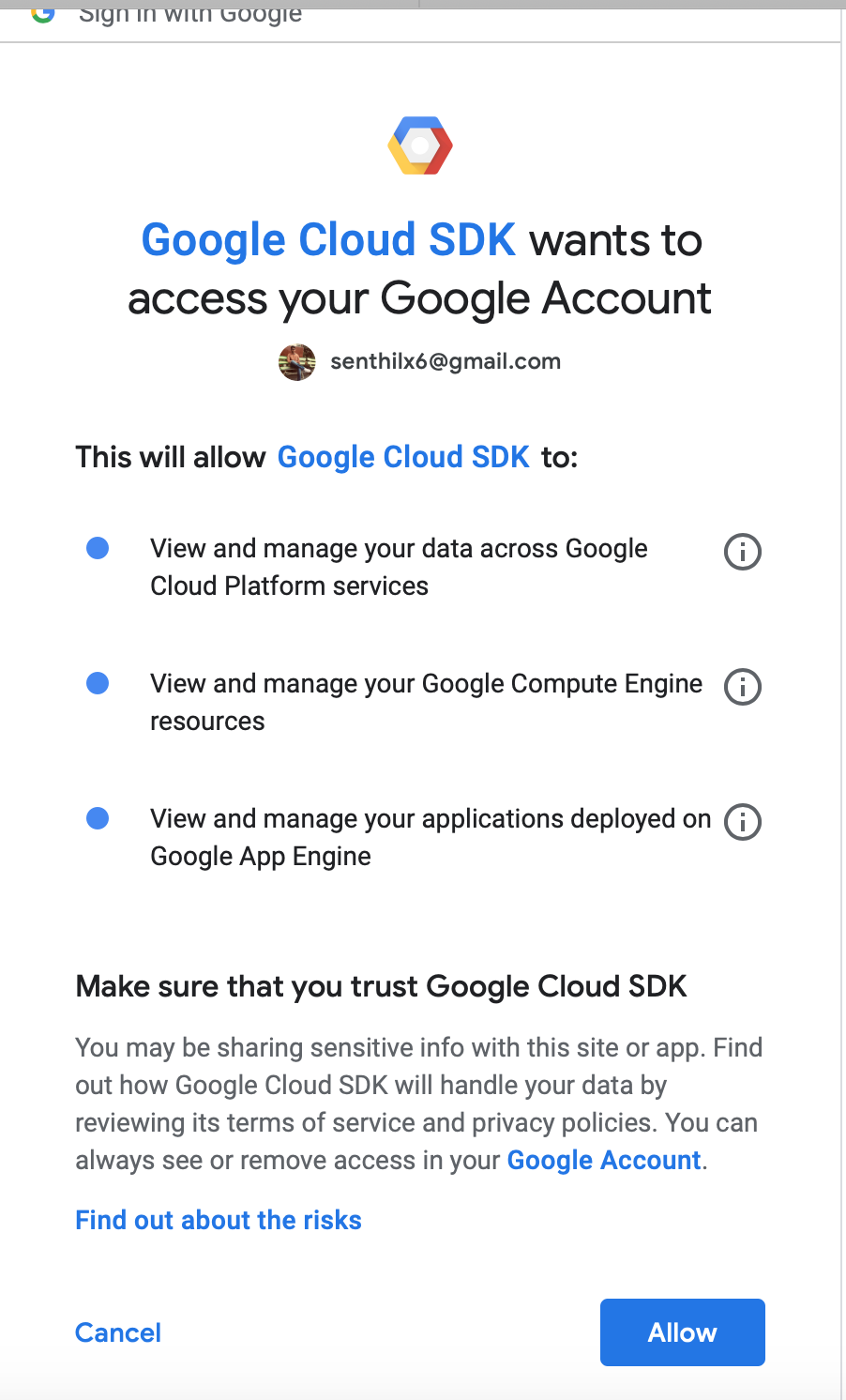


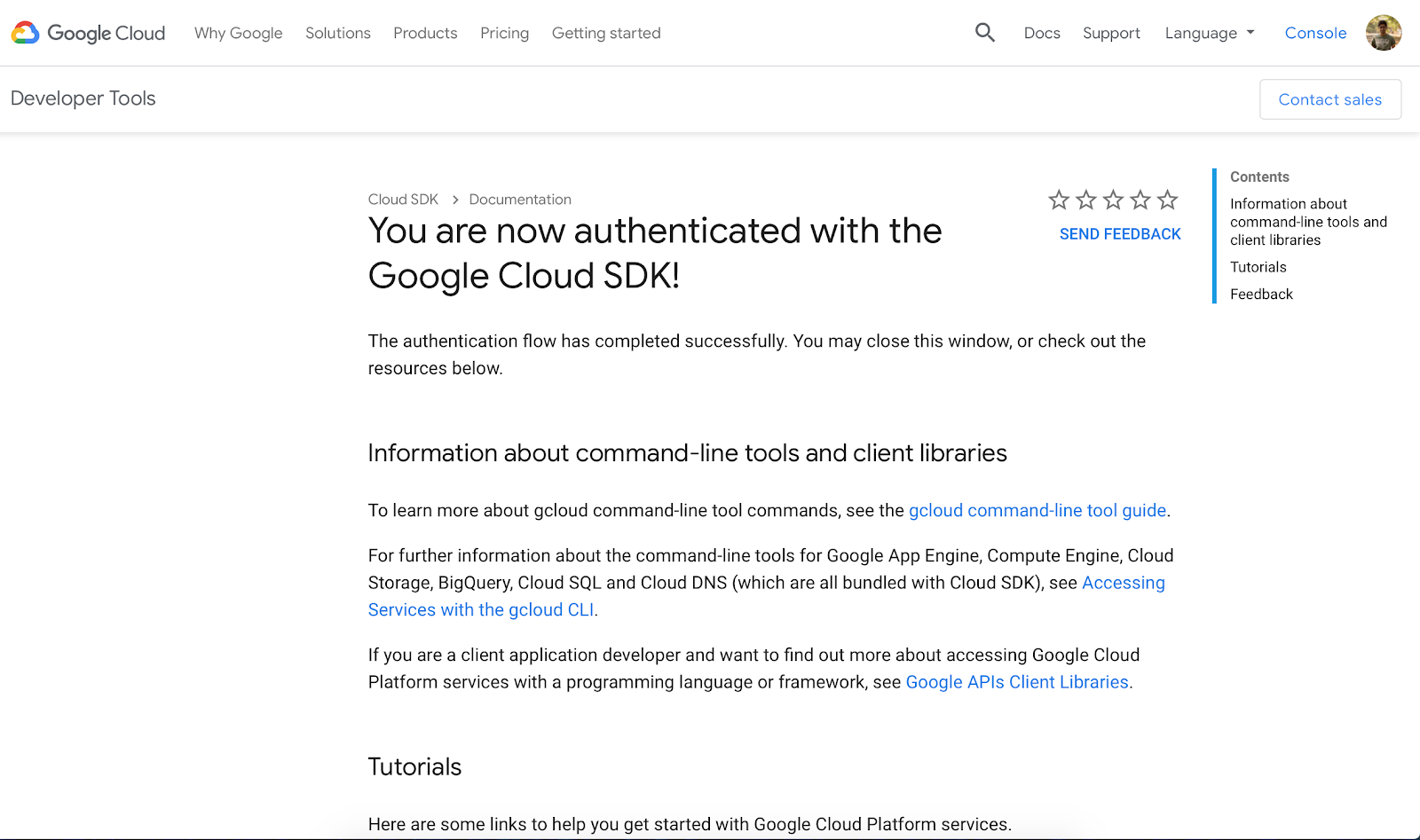
After installation is complete , Close the terminal and reopen it for the gcloud command to be activated

In the new terminal line , Give the command **gcloud init**,  so it automatically opens a browser tab asking for the google account to get authenticated

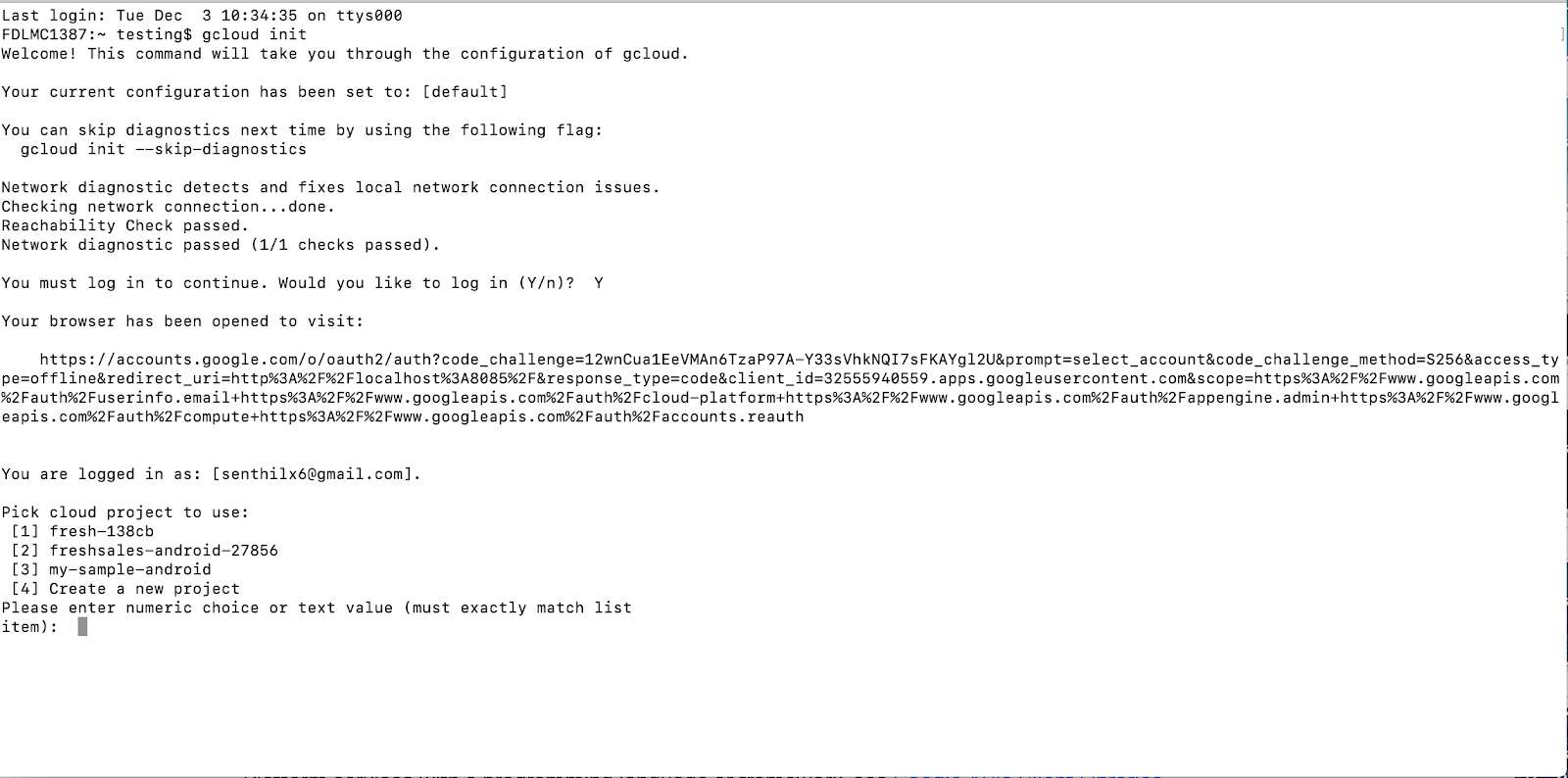








Tap Allow in the google SDK account, so that your account will be authenticated and the list of projects present in the firebase accounts are listed in the terminal

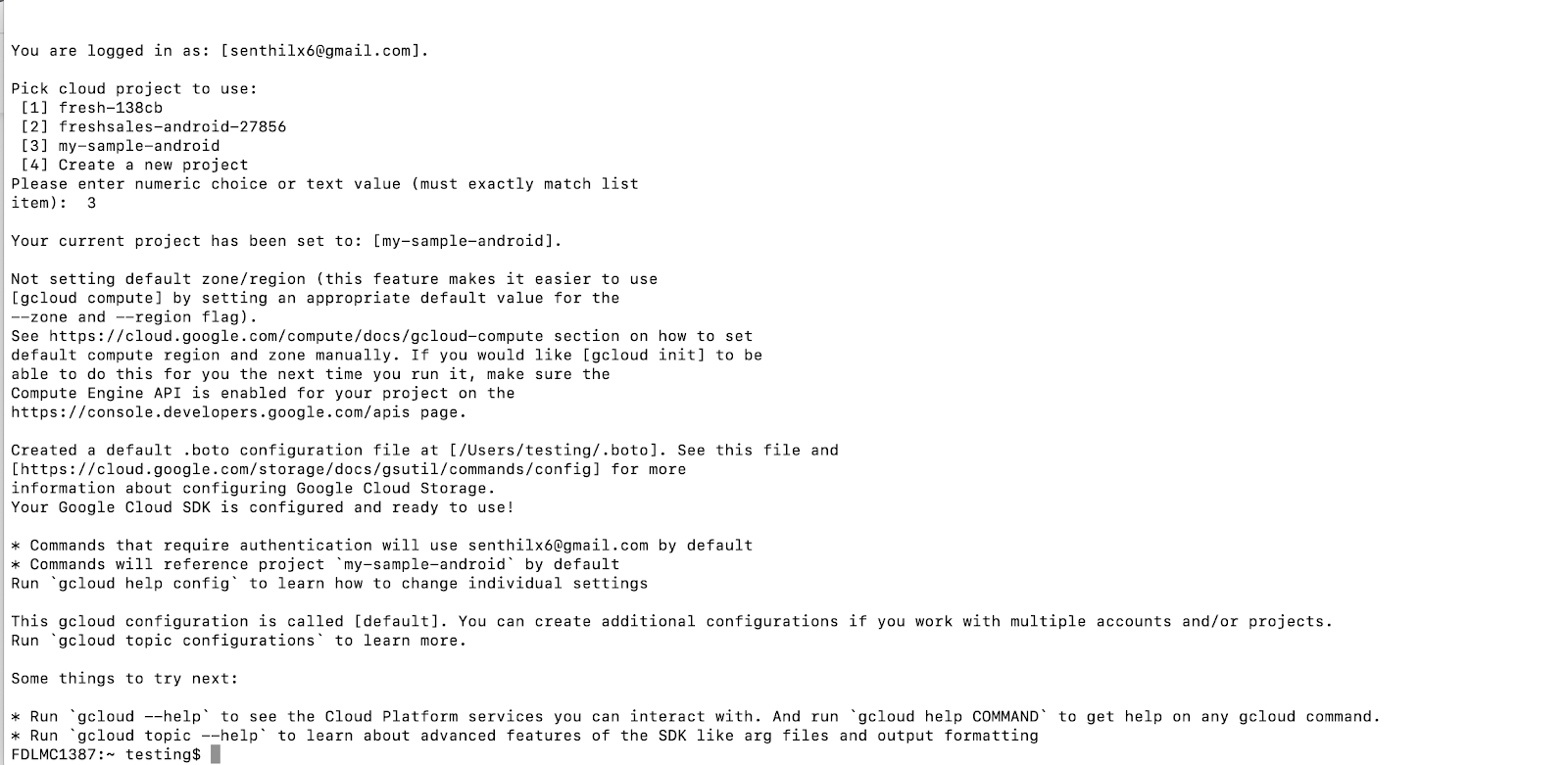


You can see the “my-sample-android” project is shown in the cloud project list

If you have more than 10 projects associated to your email id, you can try the below command

**gcloud projects list**

Where it will list all the projects, Type the number associated to the project to use the Project as default, For example, in the above terminal, we need to set my-sample-android as a default one so we type 3 which is the number associated to the my-sample-android project



The above message shown in the screenshot will be displayed once the project is associated

Now go to the android project directory  in the terminal and generate the app apk and test apk using Gradle tasks as below

**./gradlew :app:assembleDebug**

**./gradlew :app:assembleAndroidTest**

To check what are Gradle tasks present in the project directory use

**./gradlew tasks**

The above command varies to project based on the flavors and some other properties mentioned in the below link

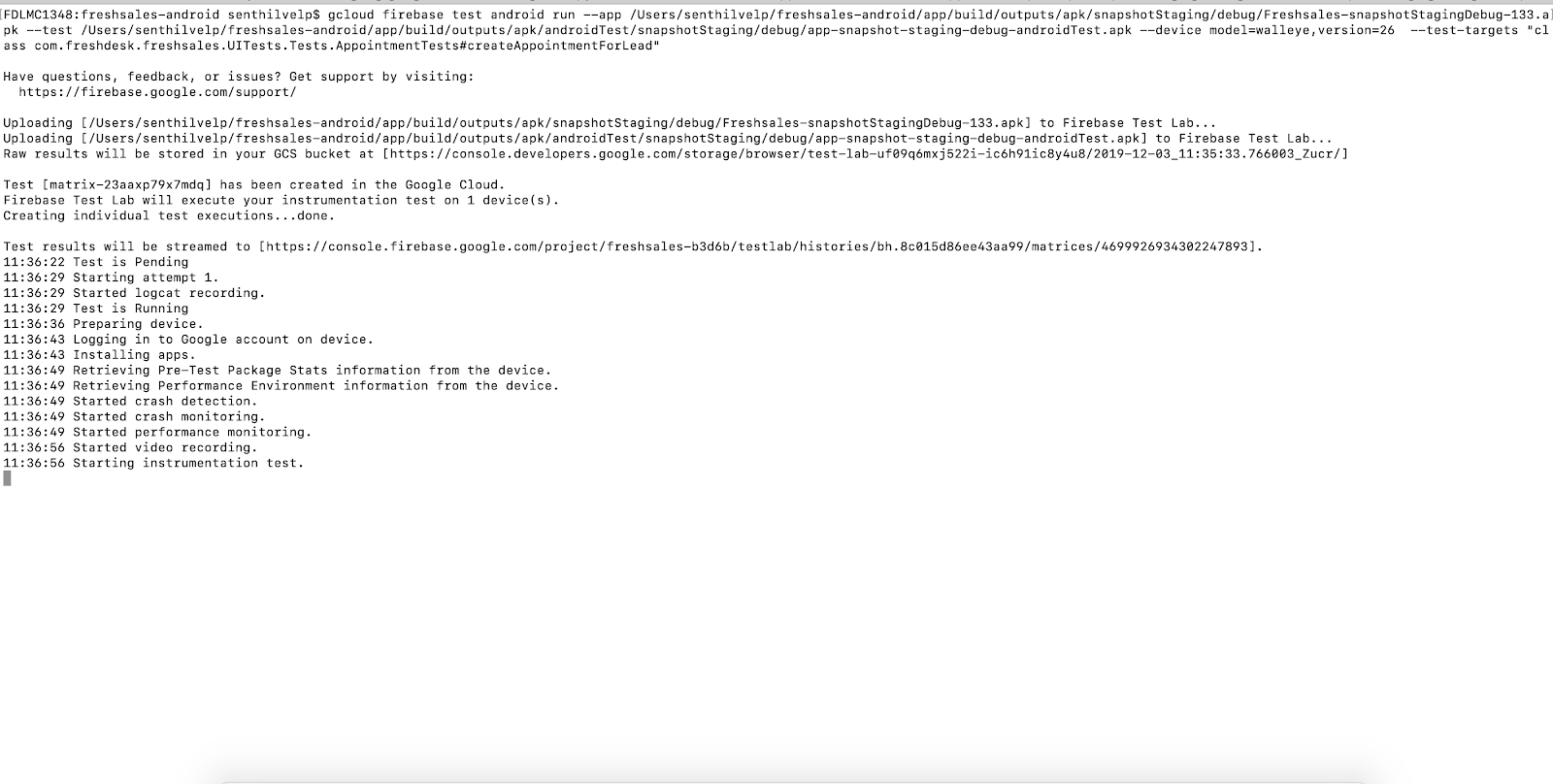
[https://developer.android.com/studio/build](https://www.google.com/url?q=https://developer.android.com/studio/build&sa=D&ust=1575459747516000) , Now the build will be generated in the folder path likeproject\_name/module\_name/build/outputs/apk

Now coming to the execution of espresso test cases via gcloud command line as below

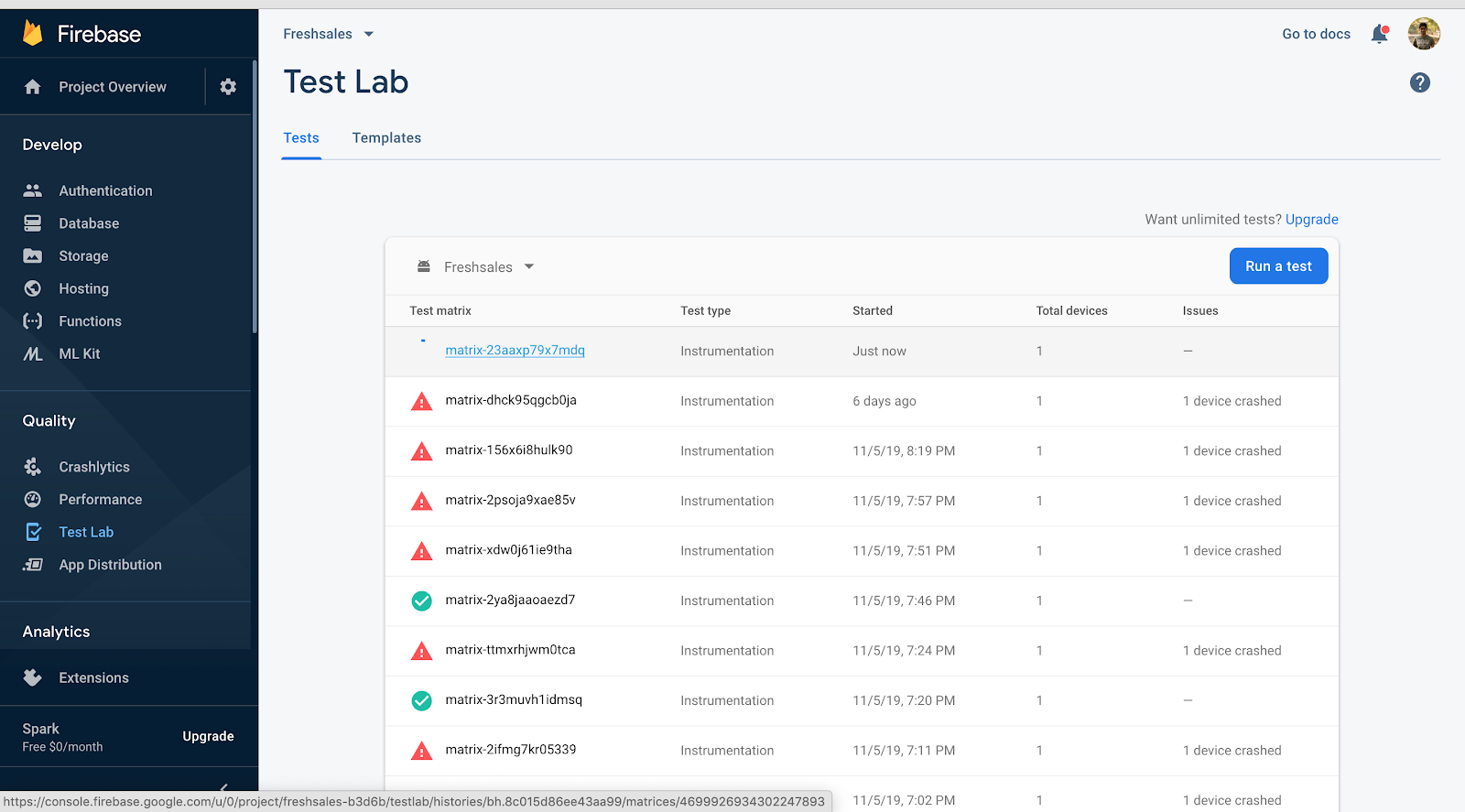
**gcloud firebase test android run \ --app path/to/app.apk \ --test path/to/tests.apk \ --device model=walleye,version=26 \ --test-targets "package** [**com.my**](http://com.my)**.package.tests1","package** [**com.my**](http://com.my)**.package.tests2"**

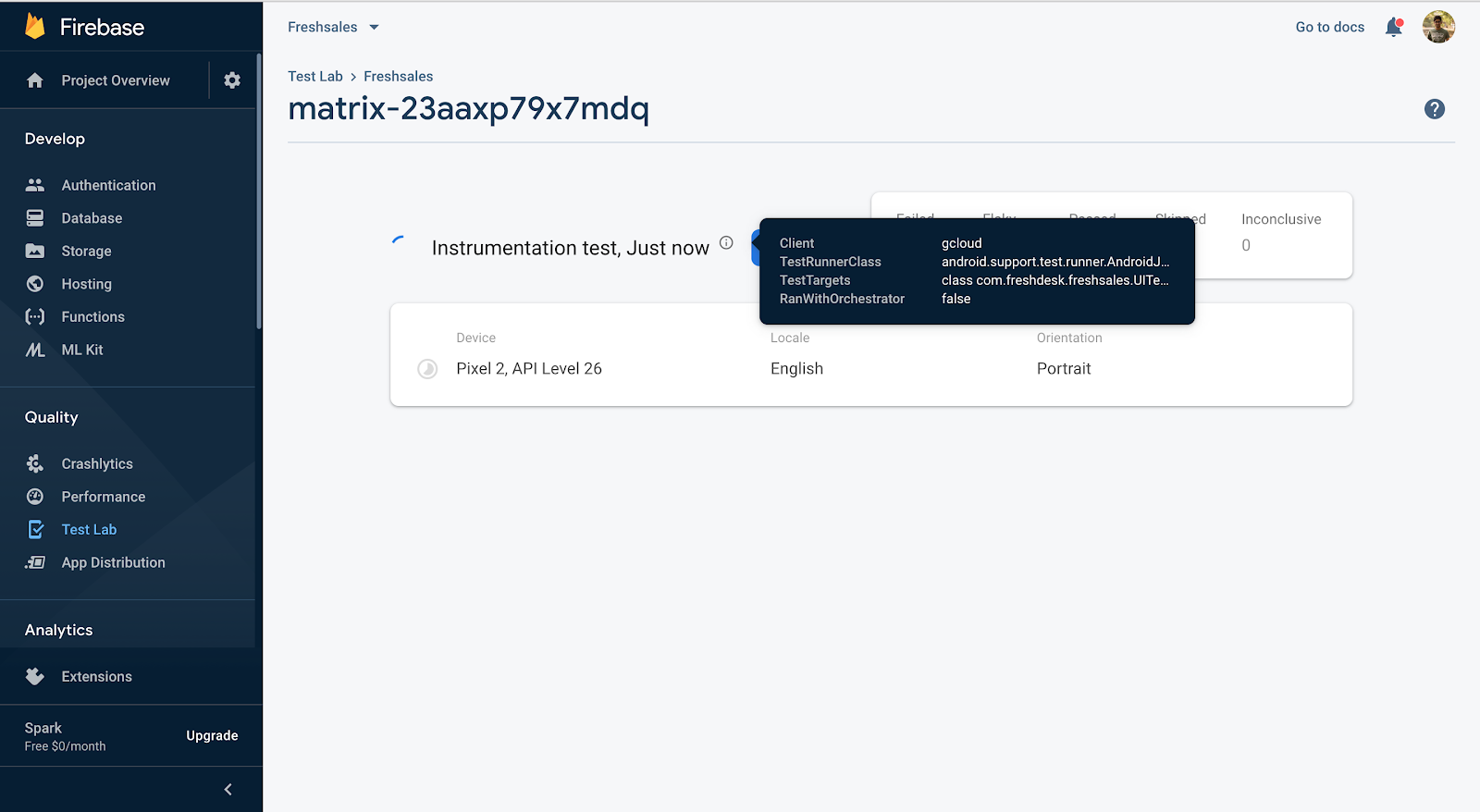
The device version can be taken from model\_id and os version id as mentioned in the glcoud firebase test android mobile list --device model=walleye,version=26

This will trigger a test case in firebase test labs as shown below



Go to your firebase project and tap on the test labs, you can find the test has been triggered and running  as shown in the screenshots and the results will be shown in the terminal and in the firebase test labs too, Firebase also displays the performance of the app during the test execution which added advantage here





### BROWSER STACK

Signup a free account in browserstack and go [https://app-automate.browserstack.com/](https://www.google.com/url?q=https://app-automate.browserstack.com/&sa=D&ust=1575459747518000) and there you can find the quick start guide in the left side panel as shown in the screenshot



Then give the below command in the terminal to upload the apk to be tested into browserstack

**curl -u "senthilvel4:sk3TeQpMDUWN8QzSZzeY" -X POST "**[**https://api-cloud.browserstack.com/app-automate/upload**](https://api-cloud.browserstack.com/app-automate/upload)**" -F "file=@/Users/senthilvelp/freshsales-android/app/build/outputs/apk/snapshotStaging/debug/Freshsales-snapshotStagingDebug-130.apk"**

This command will return an app URL  , paste it in a note pad, for example, consider this command returns the token as below

{"app\_url":"<bs://a37164545c2dc618d364aee6e445ab69f79525e9>"}

Now upload the Android test app via below command in the terminal

**curl -u "senthilvel4:sk3TeQpMDUWN8QzSZzeY" -X POST "**[**https://api-cloud.browserstack.com/app-automate/espresso/test-suite**](https://api-cloud.browserstack.com/app-automate/espresso/test-suite)**" -F "file=@/Users/senthilvelp/freshsales-android/app/build/outputs/apk/androidTest/snapshotStaging/debug/app-snapshot-staging-debug-androidTest.apk"**

This command will return an app URL, paste it in a note pad, for example, consider this command returns the token as below

{"test\_url":"<bs://1bd03f5cf1b2eb2dd492724da9bb12f15e01d8e3>"}

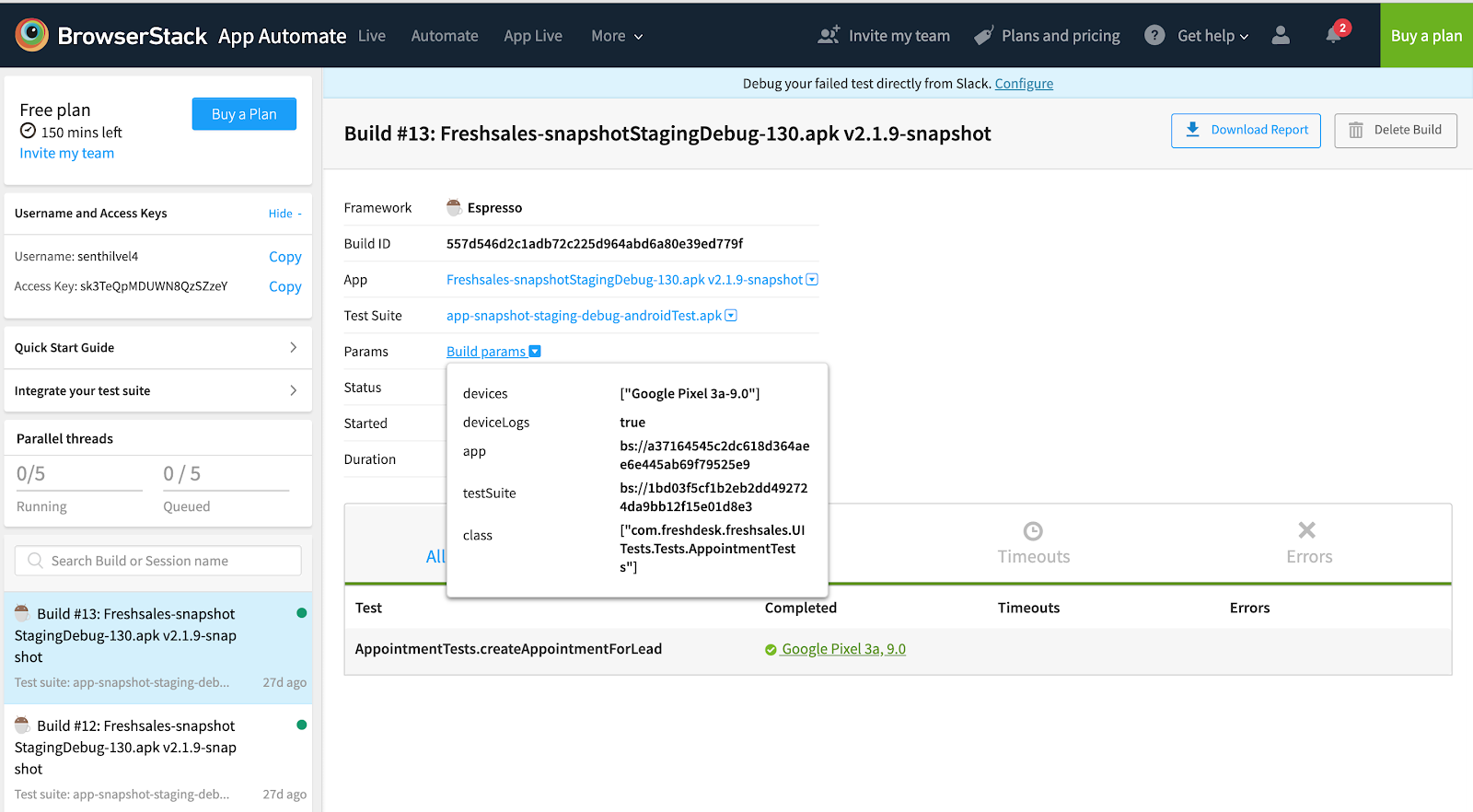
Now Run the below command in terminal to trigger a test run in the browserstack

**curl -X POST "**[**https://api-cloud.browserstack.com/app-automate/espresso/build**](https://api-cloud.browserstack.com/app-automate/espresso/build)**" -d \ "{\"devices\": [\"Google Pixel 3a-9.0\"], \"app\": \"**[**bs://a37164545c2dc618d364aee6e445ab69f79525e9\**](bs://a37164545c2dc618d364aee6e445ab69f79525e9\)**", \"deviceLogs\" : true,\"class\" : [\"com.freshdesk.freshsales.UITests.Tests.AppointmentTests\"], \"testSuite\": \"**[**bs://1bd03f5cf1b2eb2dd492724da9bb12f15e01d8e3\**](bs://1bd03f5cf1b2eb2dd492724da9bb12f15e01d8e3\)**"}" -H "Content-Type: application/json" -u "senthilvel4:sk3TeQpMDUWN8QzSZzeY"**

**Note:**

Now you can see test triggered in the browserstack account with test class shown below, you can run the test based on test suite, package, class, annotation, for more reference please refer the below link

[https://www.browserstack.com/app-automate/espresso/get-started](https://www.google.com/url?q=https://www.browserstack.com/app-automate/espresso/get-started&sa=D&ust=1575459747521000)



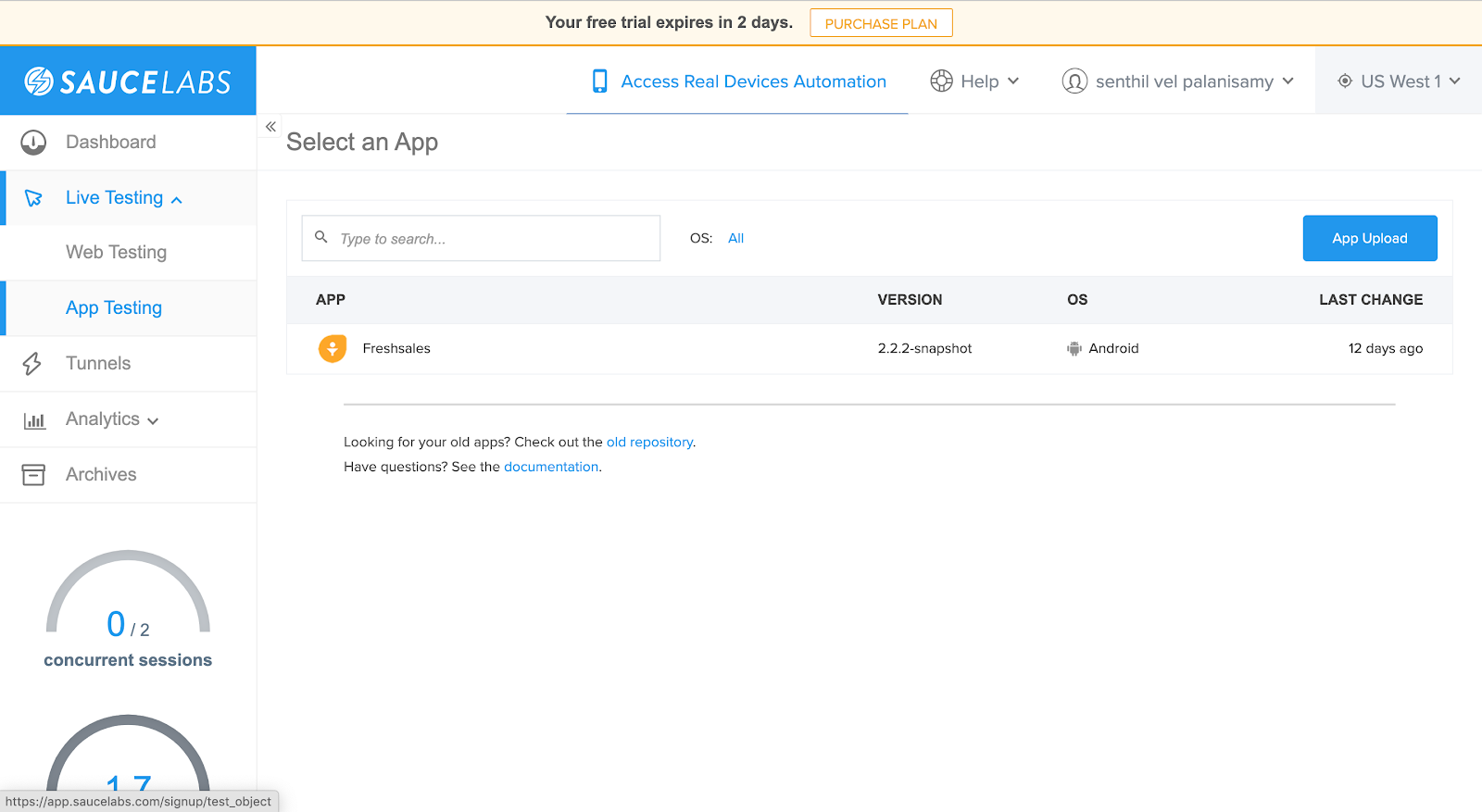
**Note:**

**For user name and Access key, you can find the same in the left panel shown in the above screenshot**

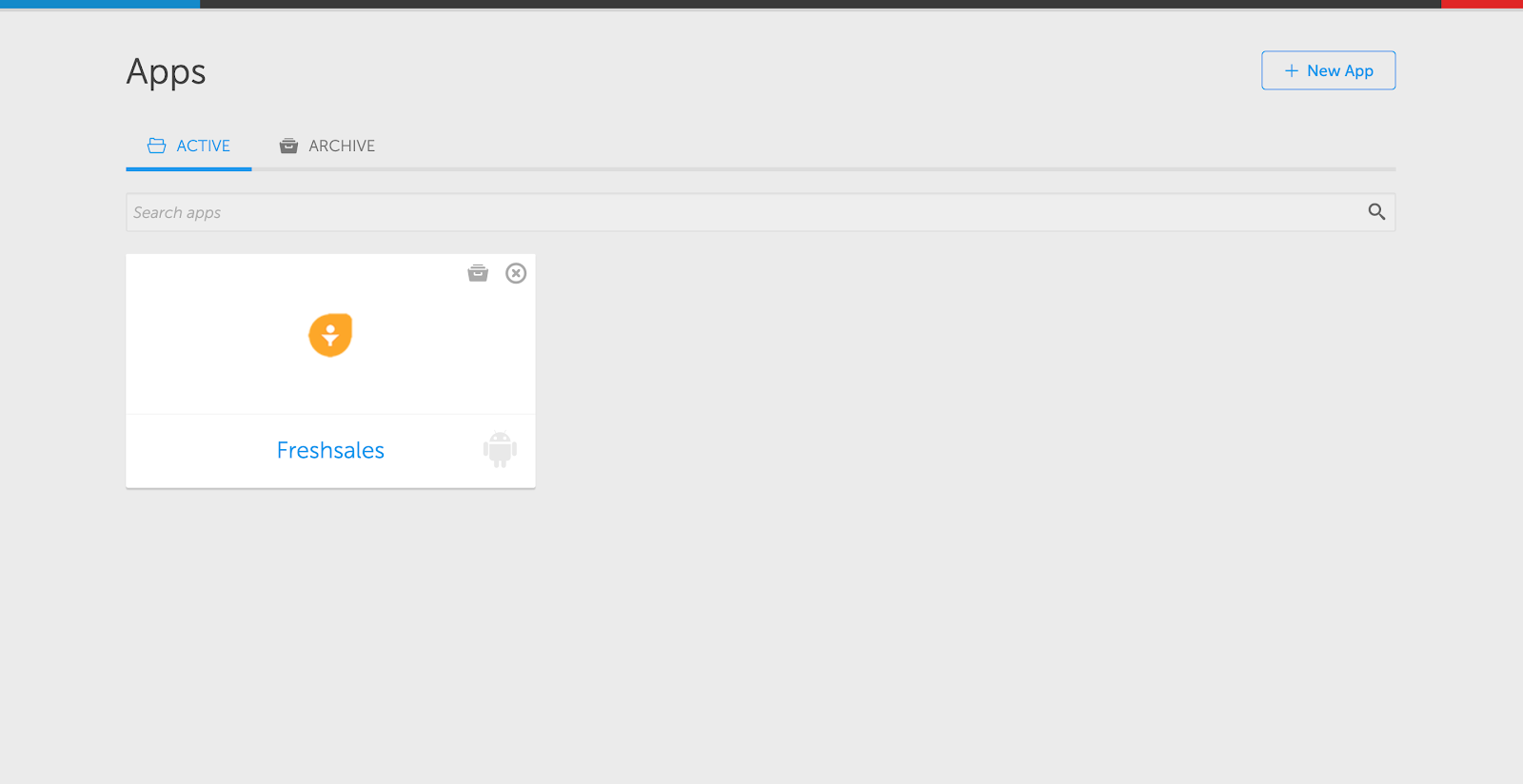
## **SauceLabs**

Signup a free account in sauce labs in the below URL

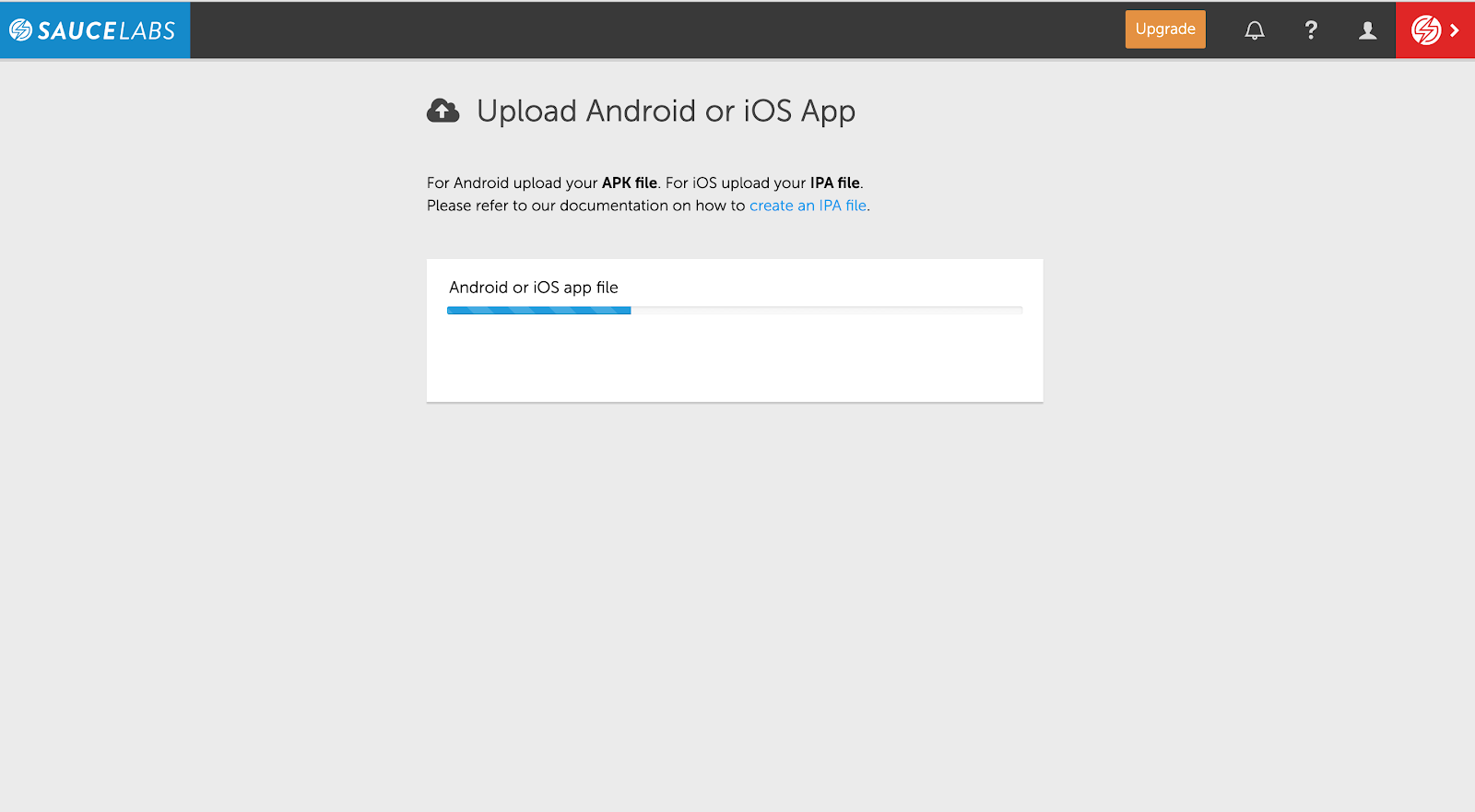
[https://saucelabs.com/sign-up](https://www.google.com/url?q=https://saucelabs.com/sign-up&sa=D&ust=1575459747524000) and after successful sign up navigate to [https://app.saucelabs.com/](https://www.google.com/url?q=https://app.saucelabs.com/&sa=D&ust=1575459747524000) , there you can see Access real device automation as shown in the below screenshot, tap on the Access real device automation and it will navigate to [https://app.testobject.com/](https://www.google.com/url?q=https://app.testobject.com/&sa=D&ust=1575459747525000)



In that URL please create a new app  using tapping on the New App Button as shown in the below

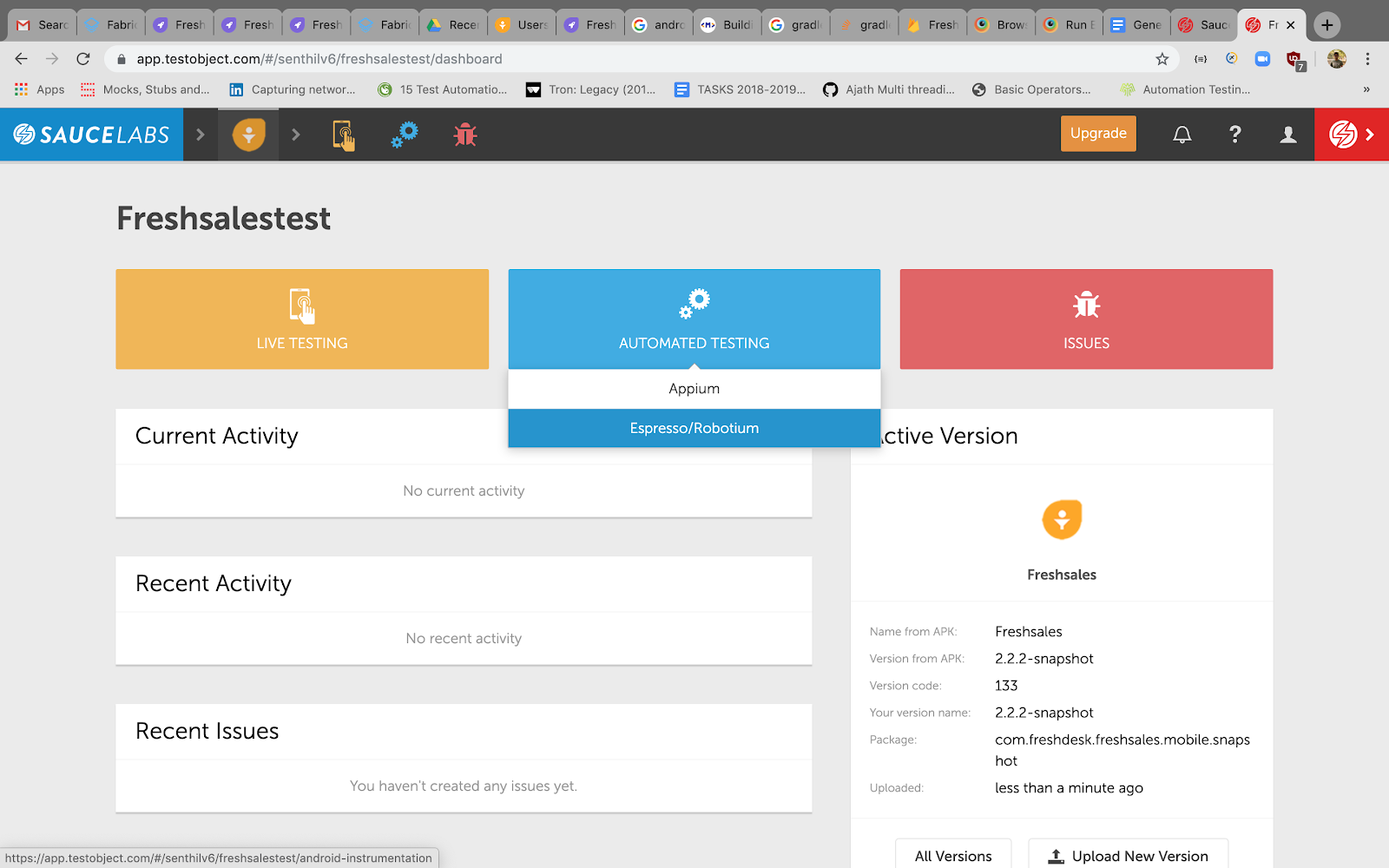


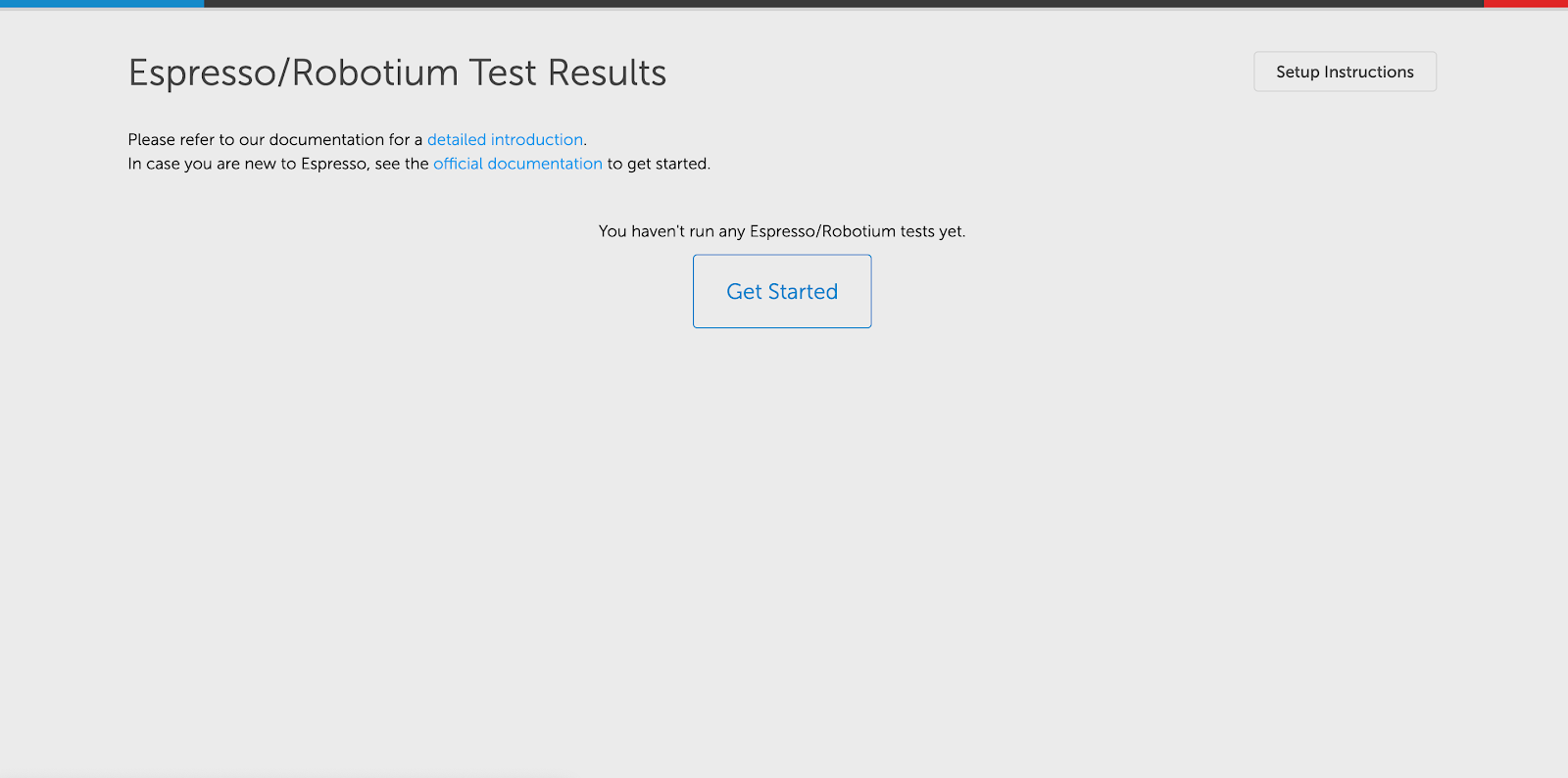
Tap on the Android IOS App in the next page , this will ask the App to be tested to get uploaded , please upload the app here



Then it will ask for the device orientation, give the default orientation and tap save

After this , the project will be created and for executing espresso test tap on the below icon shown in the screenshot

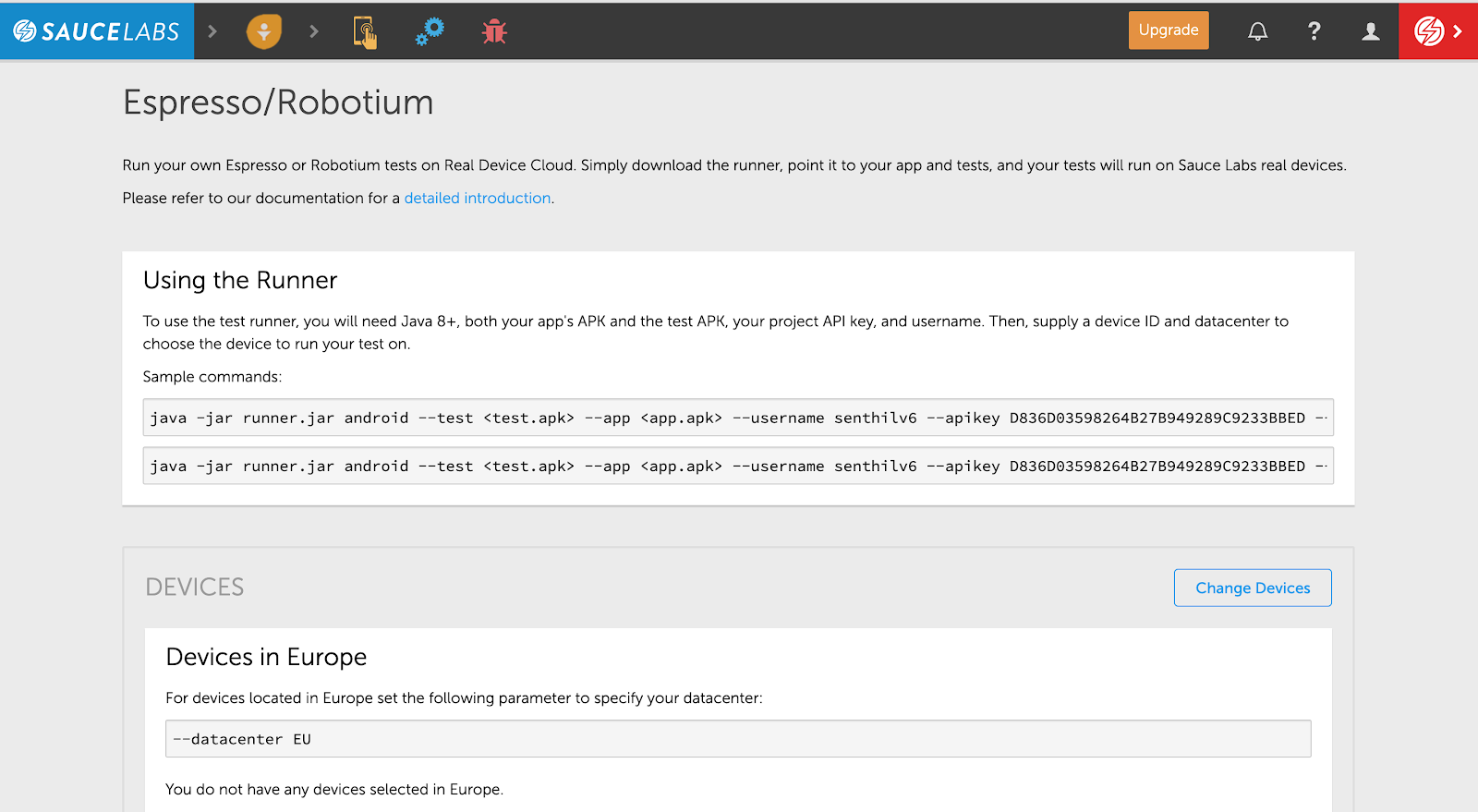




For running espresso test we need to download the runner.java file provided by the sauce labs from the below link

[https://wiki.saucelabs.com/display/DOCS/Using+Espresso+for+Real+Device+Testing](https://www.google.com/url?q=https://wiki.saucelabs.com/display/DOCS/Using%2BEspresso%2Bfor%2BReal%2BDevice%2BTesting&sa=D&ust=1575459747528000)

After downloading the jar, copy and paste in your home directory, to know the sauce labs secret key , Tap on the Get started button in the above screenshot and this will take you to below screen

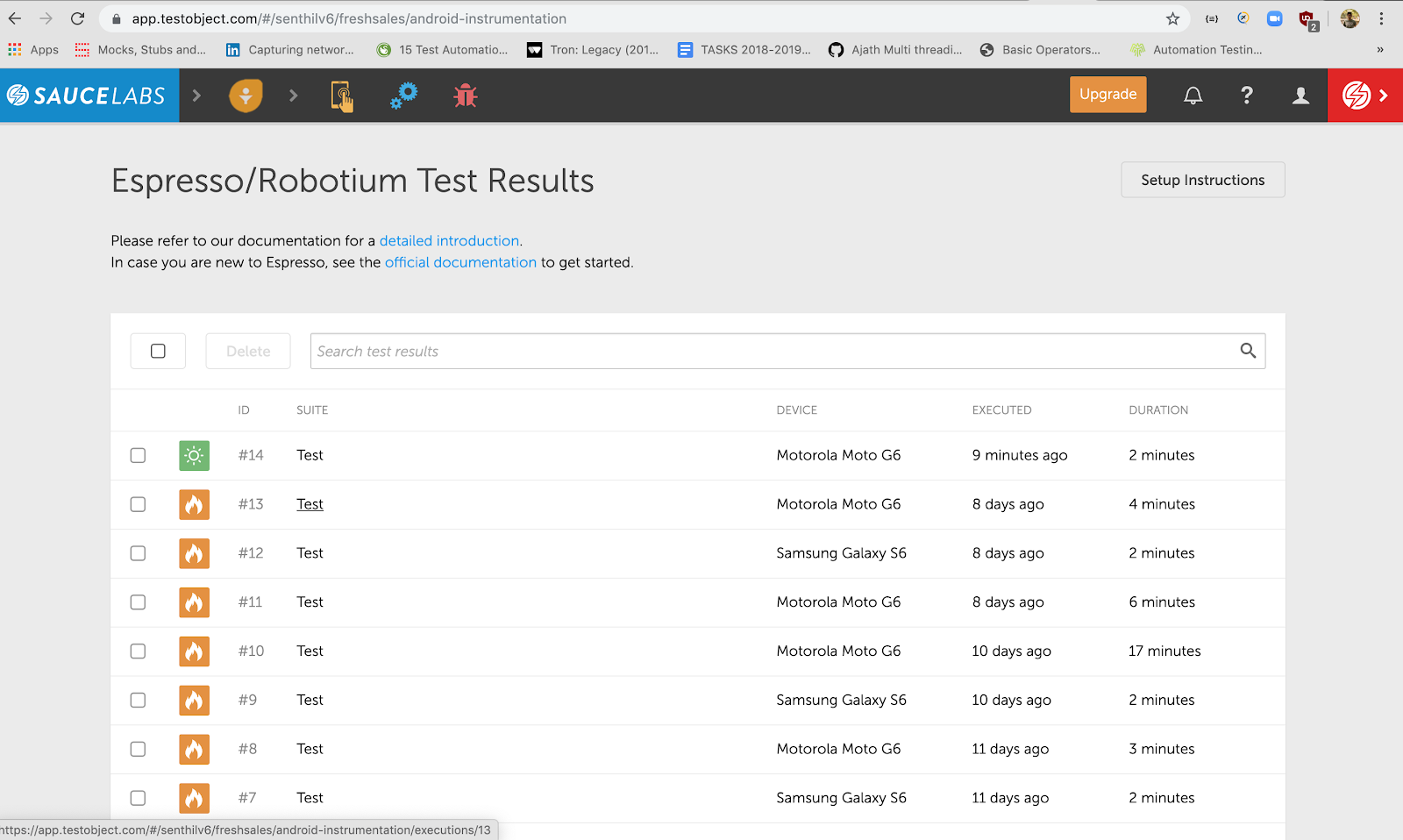


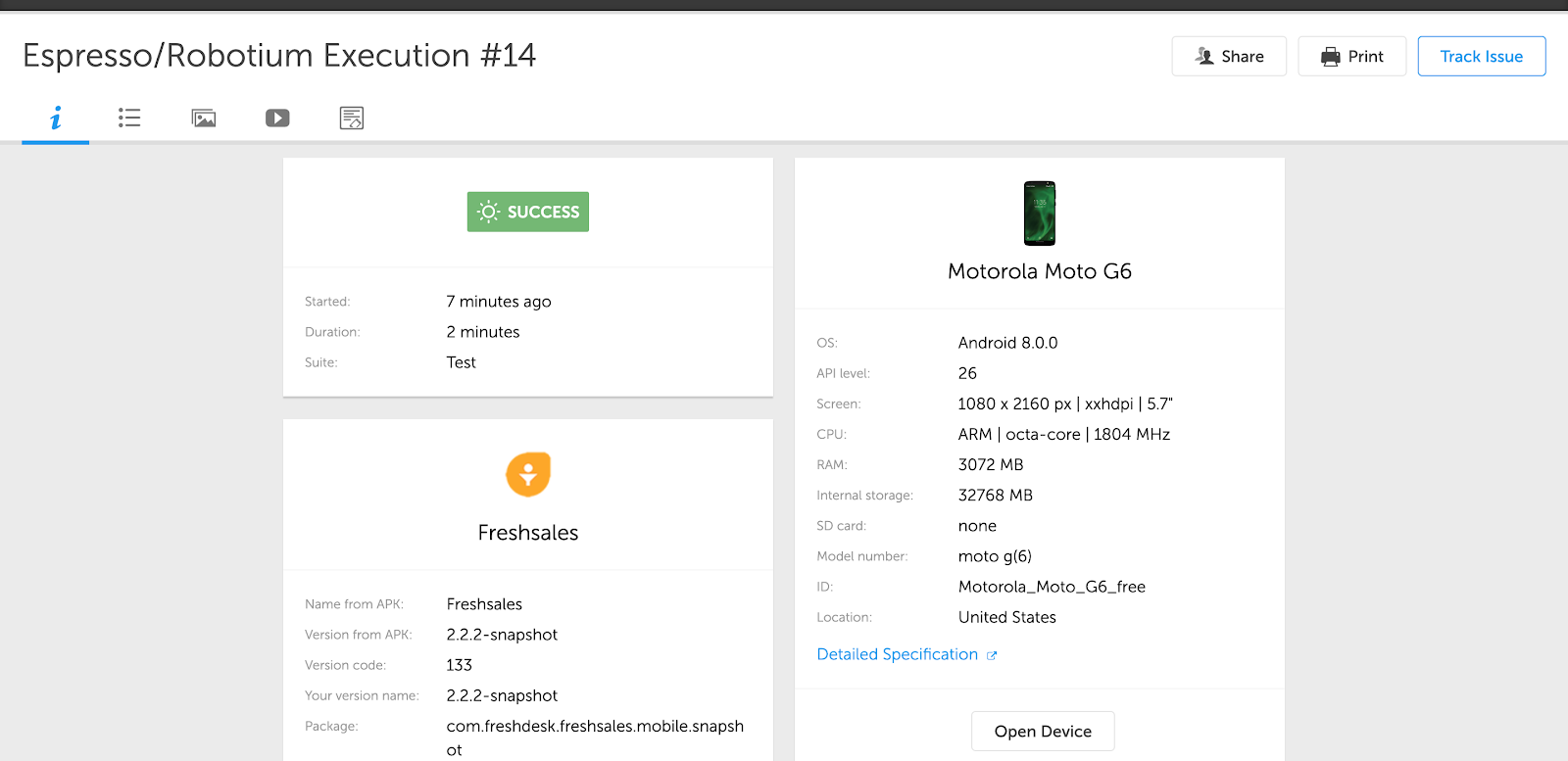
Now got home directory where the runner.jave is located in the terminal and type the below command

**java -jar runner.jar espresso --test /Users/senthilvelp/freshsales-android/app/build/outputs/apk/androidTest/snapshotStaging/debug/app-snapshot-staging-debug-androidTest.apk --app /Users/senthilvelp/freshsales-android/app/build/outputs/apk/snapshotStaging/debug/Freshsales-snapshotStagingDebug-133.apk --apikey 0E4703278D544FD1AD7B79873252F3C1 --datacenter US --e class com.freshdesk.freshsales.UITests.Tests.AppointmentTests#createAppointmentForLead**

This will trigger a test in sauce labs

Then refresh the page you can see the test suite got triggered or go to the project dashboard and click automated testing → espresso / robotium and you can find the test suite as below





**Sauce labs supports test execution based on class, package, and annotation too with help of the -e command**