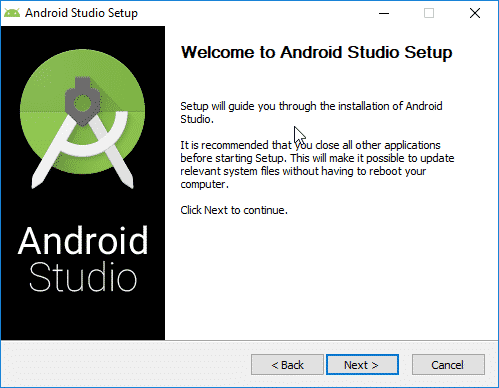
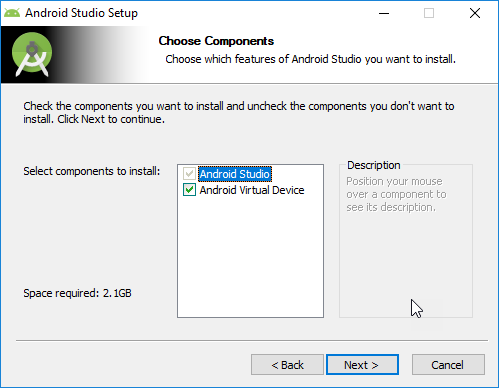
**Android Studio Installation**

**Step 1)**Go to<https://developer.android.com/studio> and click on Android Studio.

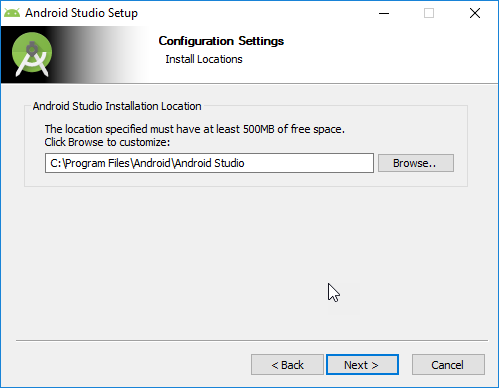
**Step 2)** Click the next button.



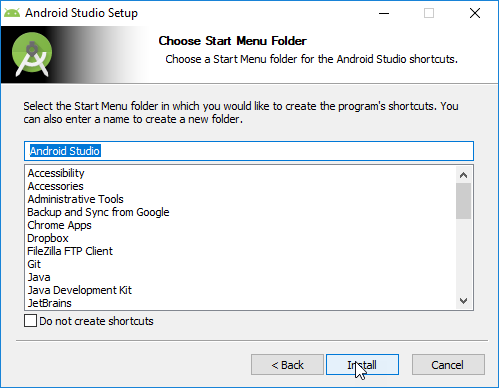
**Step 3)** Select all options and click the next button.



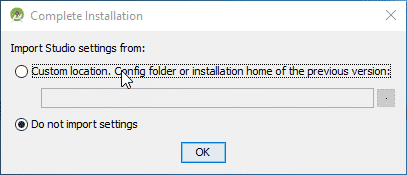
**Step 4)** Select your directory or keep the default and click next.



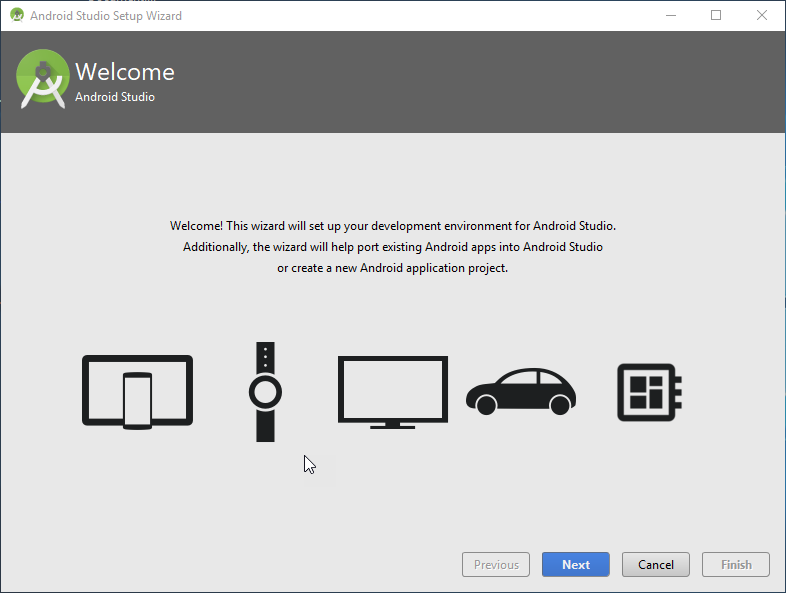
**Step 5)** and click install to start the installation process.



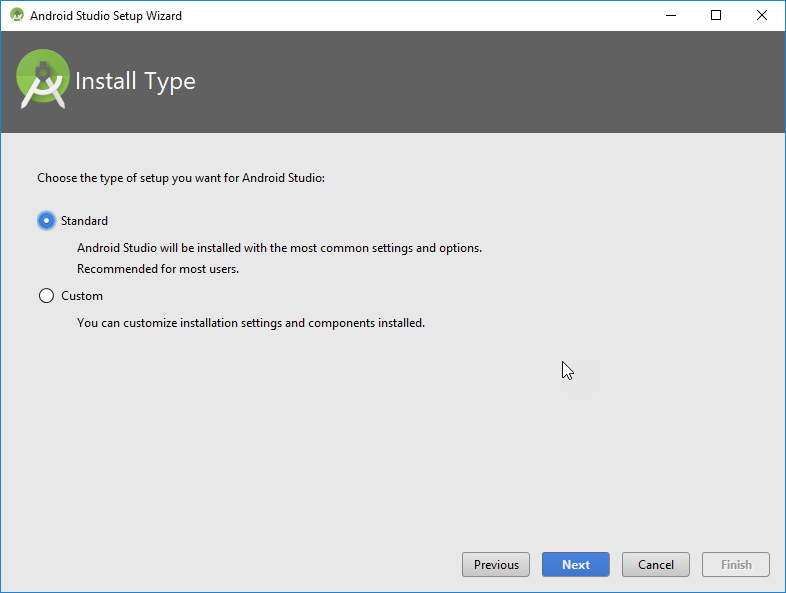
**Step 6)** It is my first time to install Android Studio and I selected “Do not import settings” option and clicked the OK button.



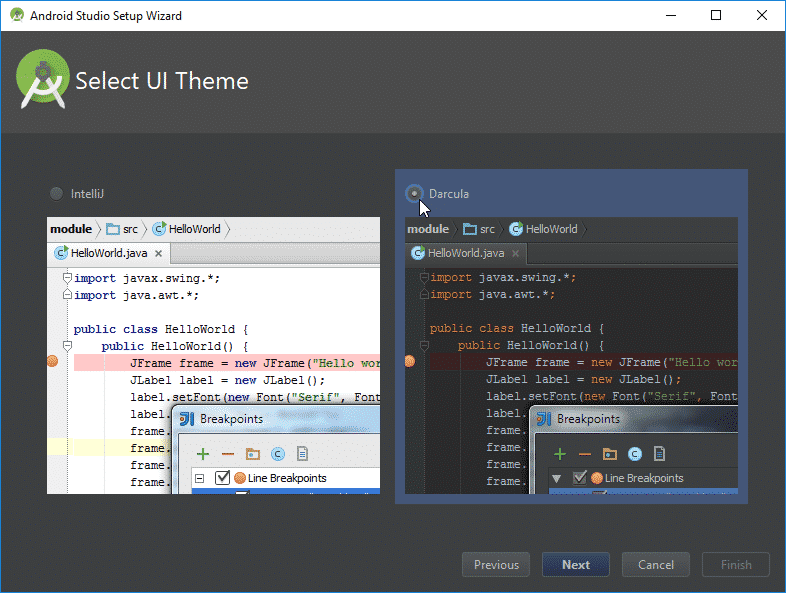
**Step 7)** Click next to continue.



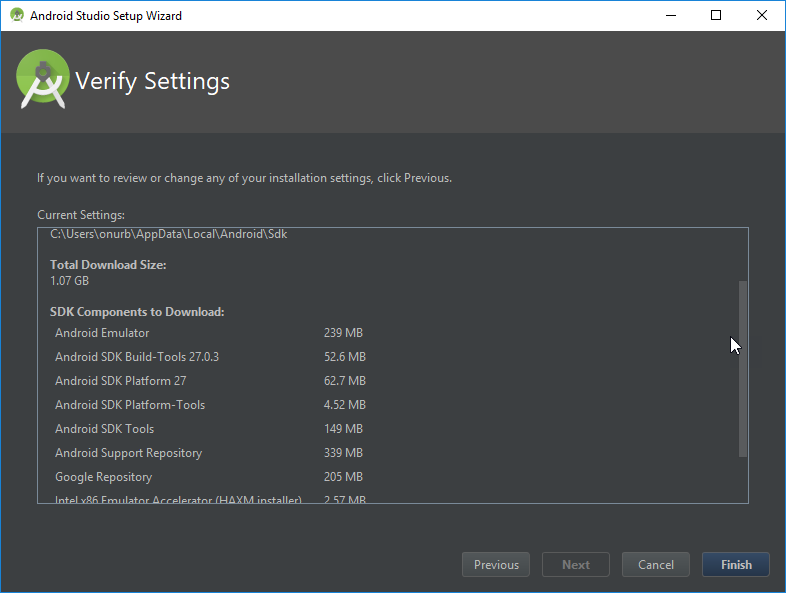
**Step 8)** I installed the Android Studio with standard settings and configurations.



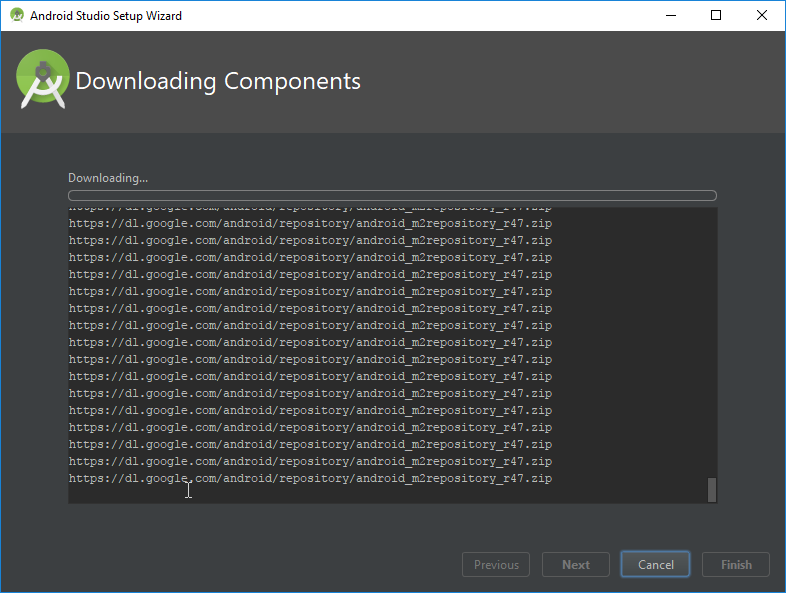
**Step 9)** Select your UI Theme.



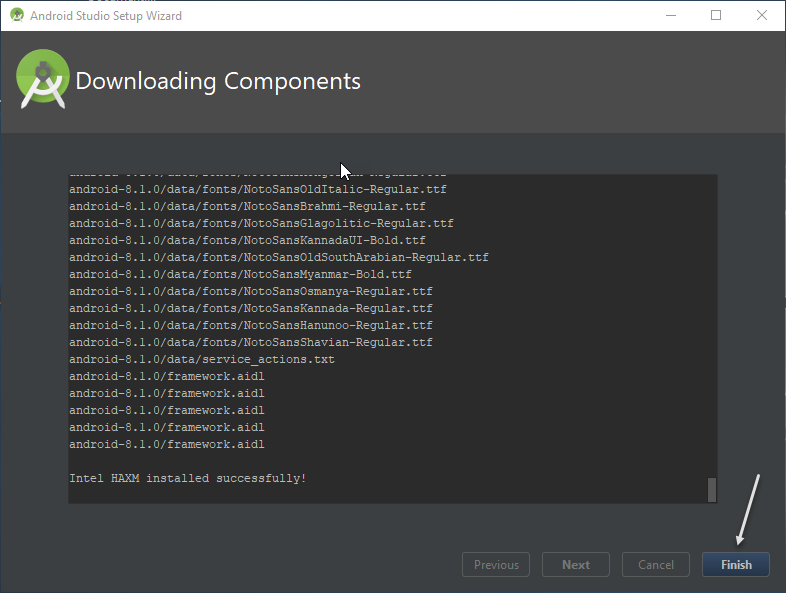
**Step 10)** Click “Finish” button and install the required tools.



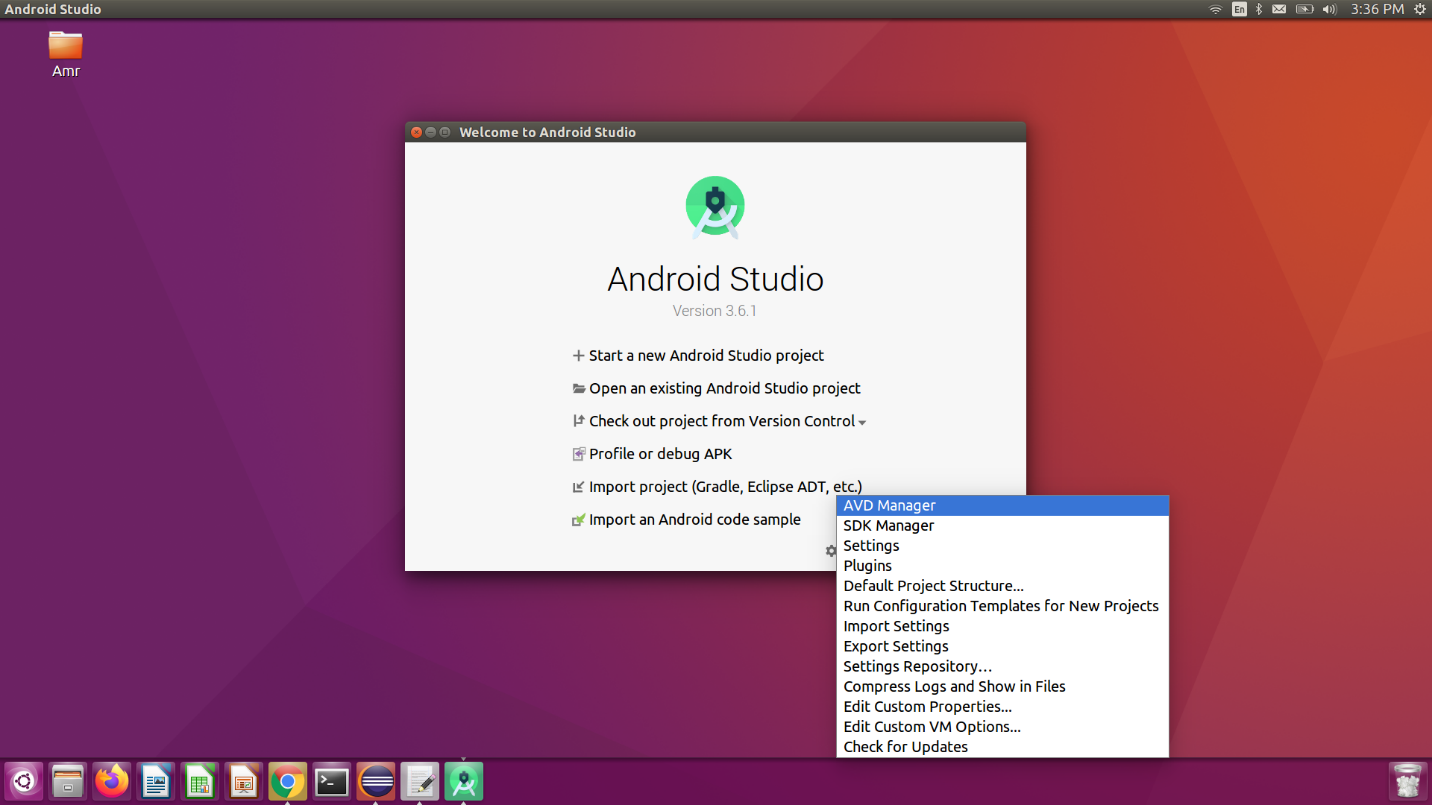
**Step 11)** Wait until all libraries download and installation finish.



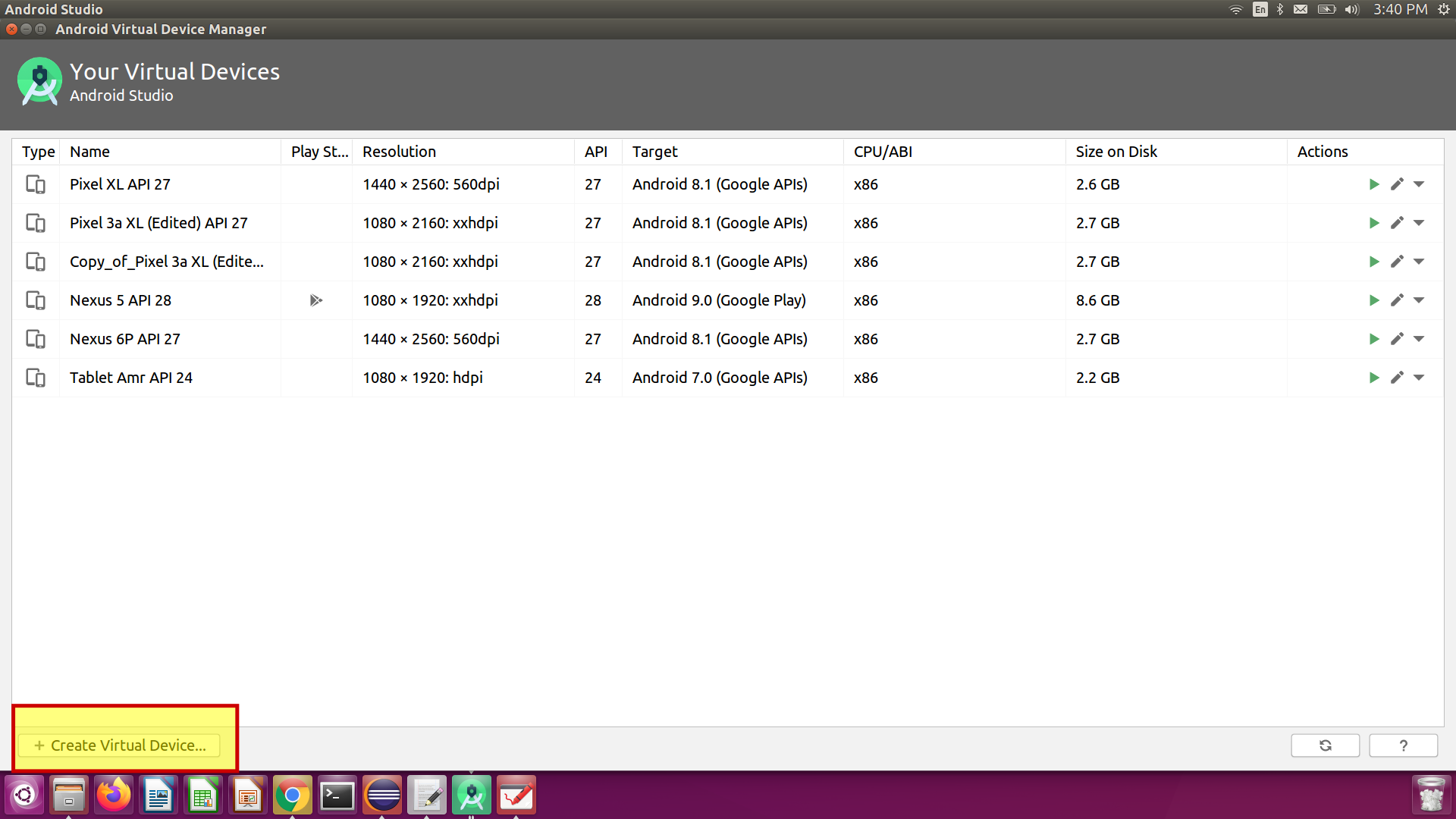
**Step 12)** After all these steps, click the finish button.



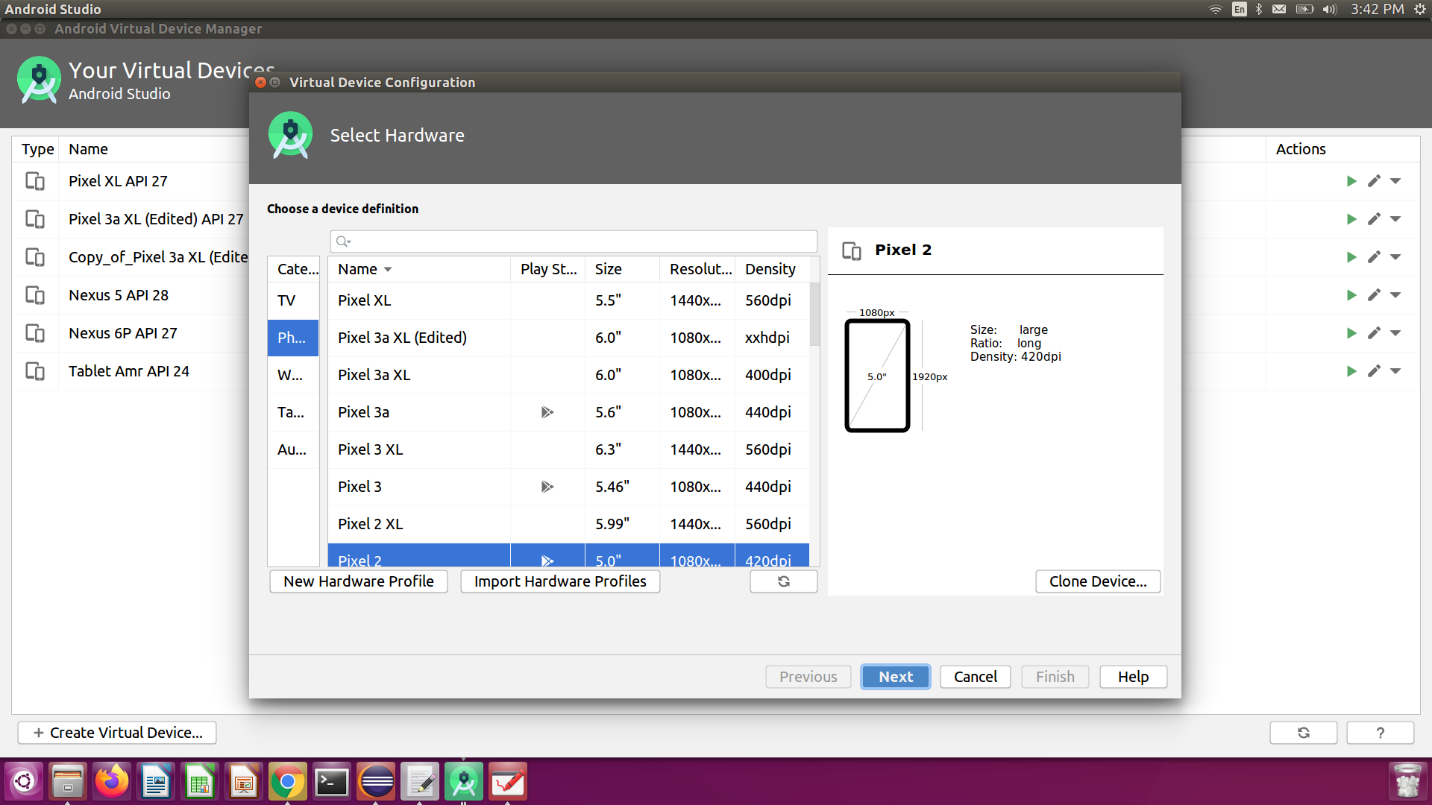
**Step 13)** After clicking the “Finish” button. Go to “Configure” > “AVD Manager”



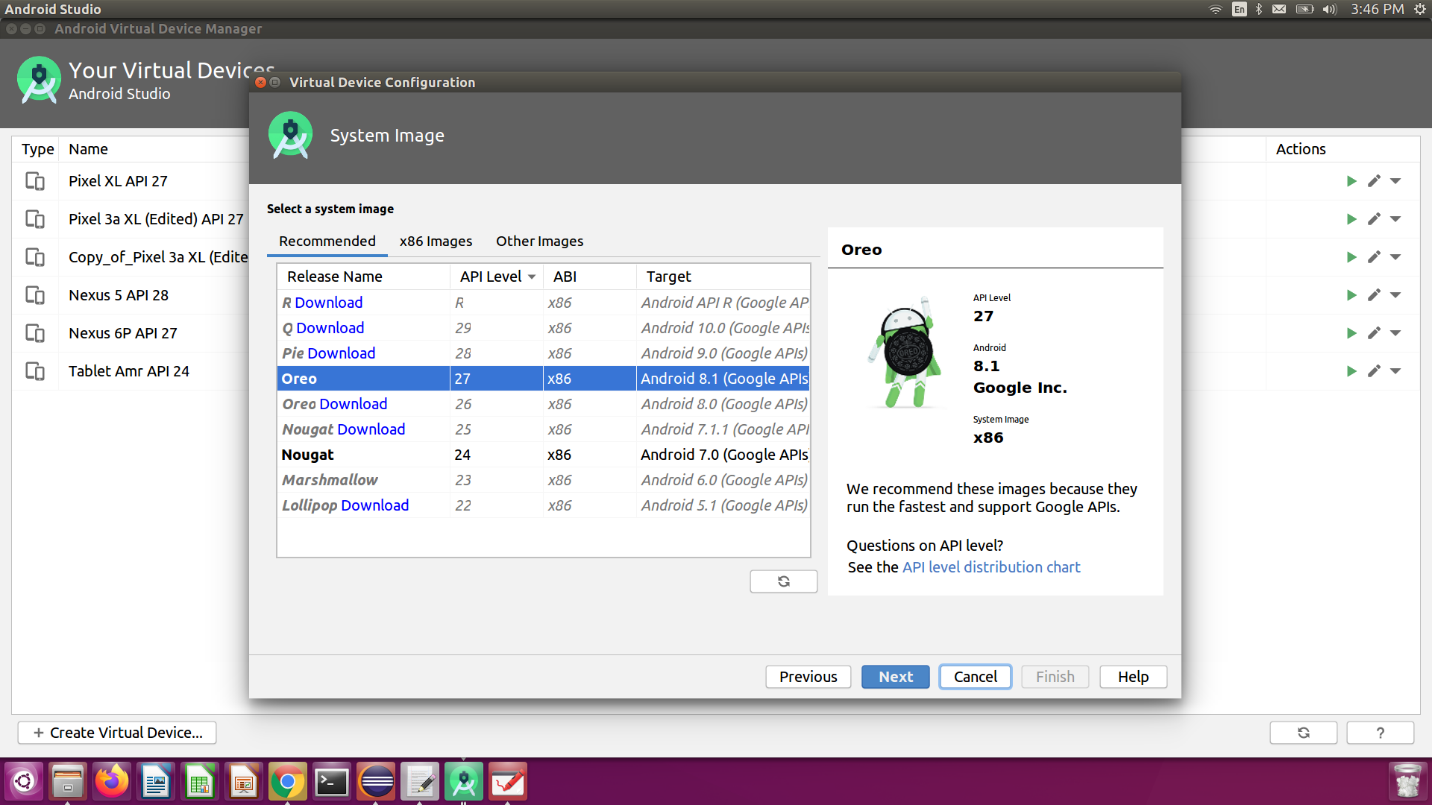
**Step 14)** Click "Create Virtuial Device"



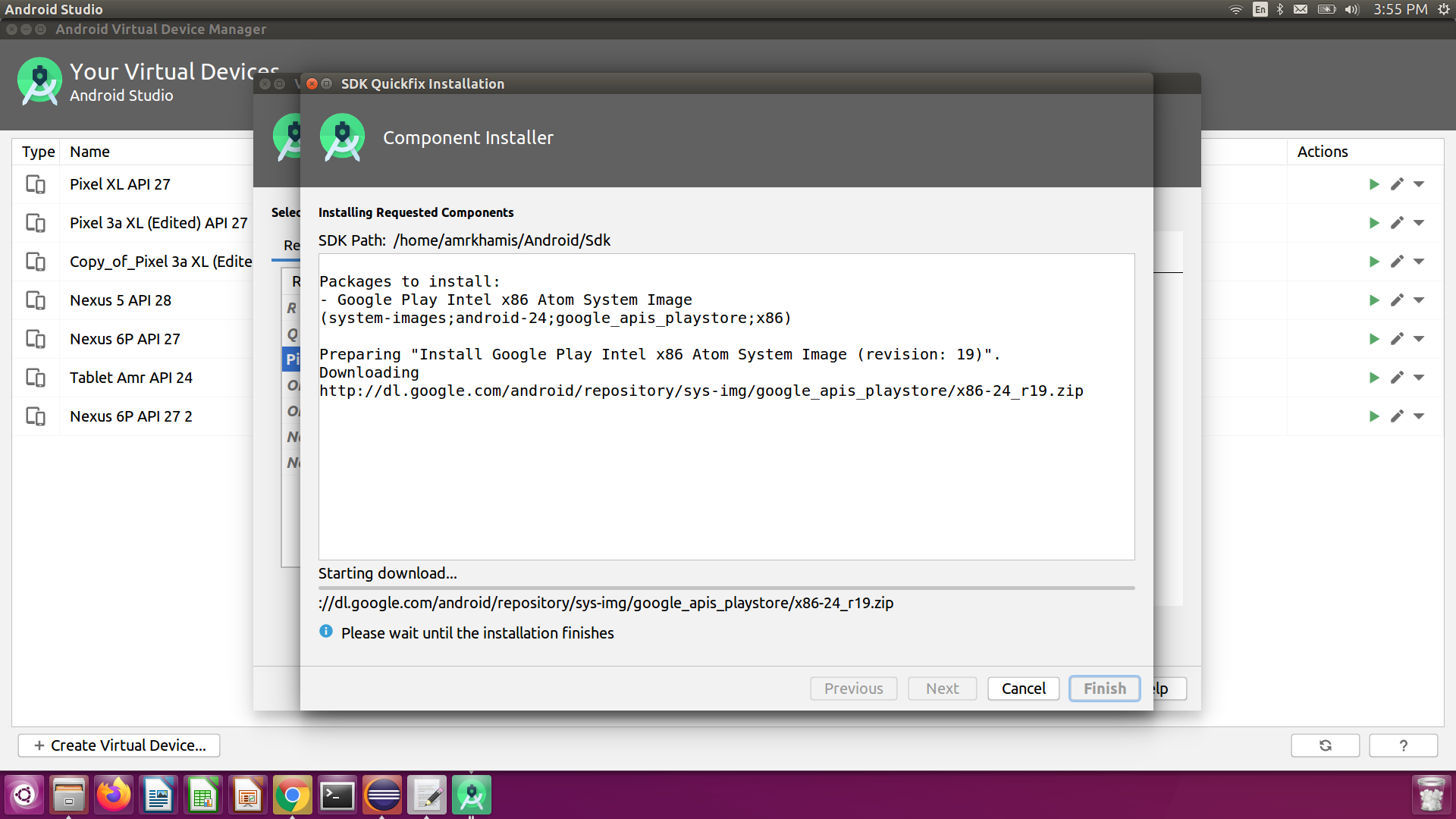
**Step 15)** Select Hardware and click "Next" button



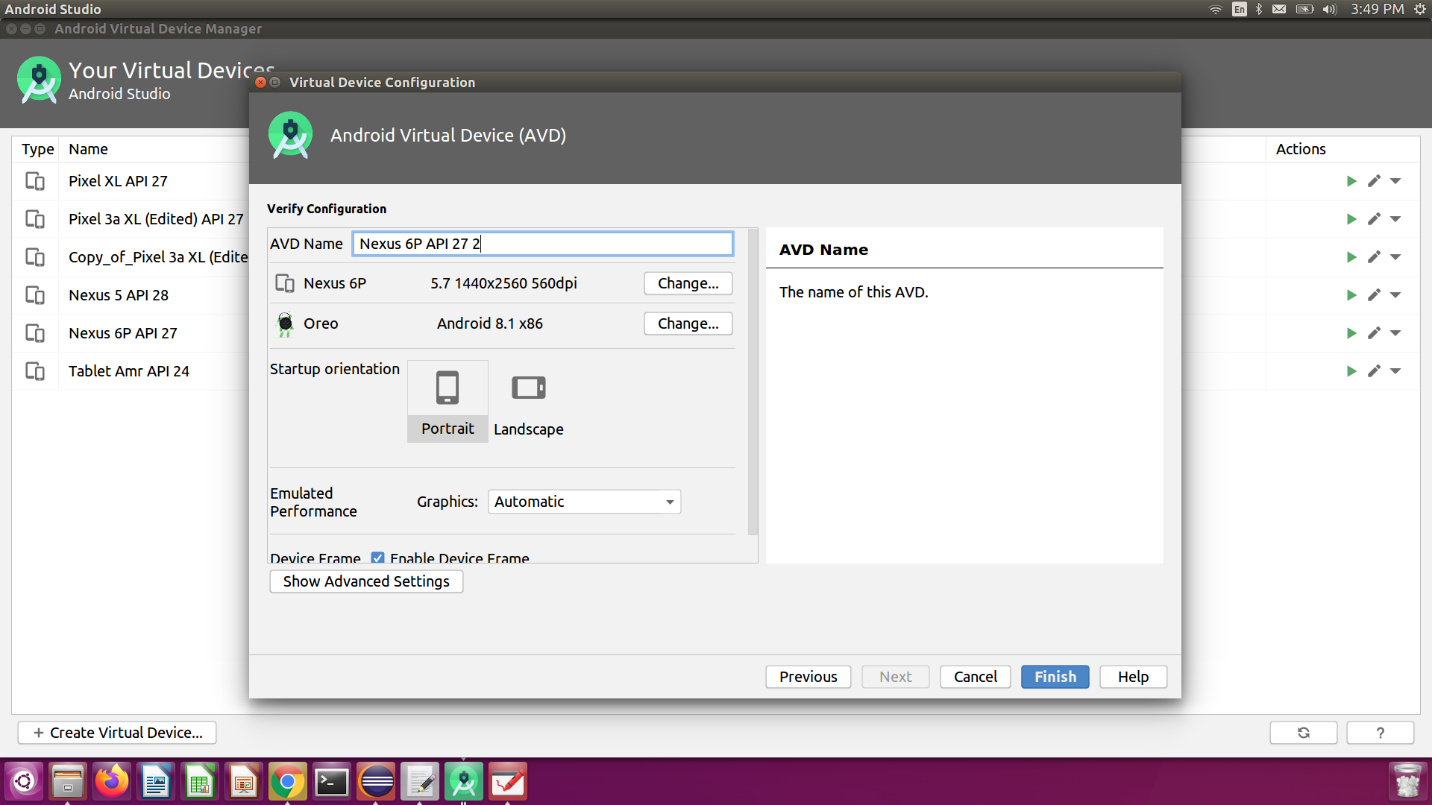
**Step 16)**Download system image



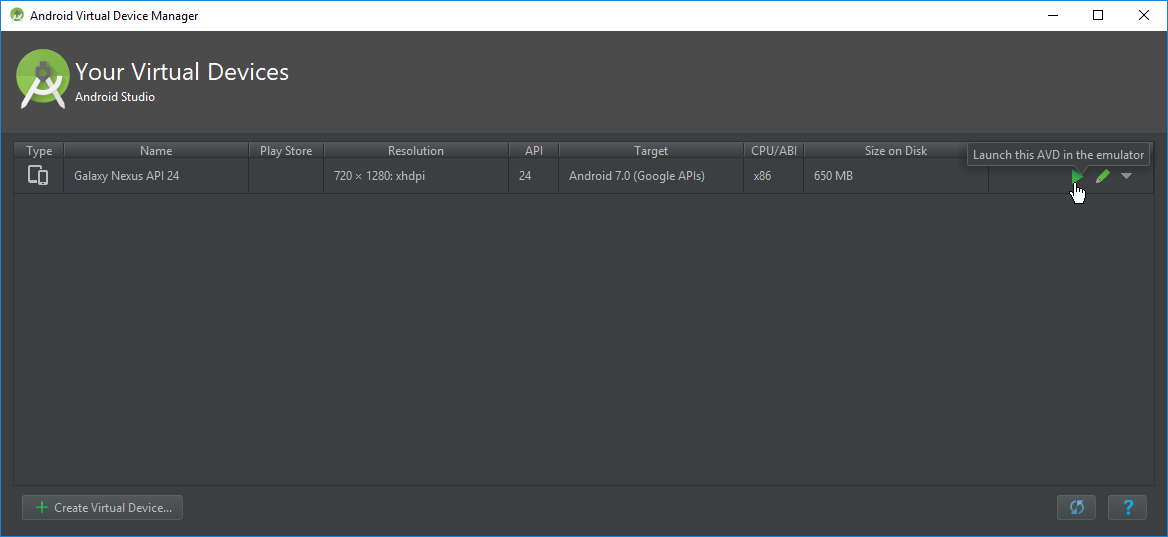
**Step 17)**Wait until component installed



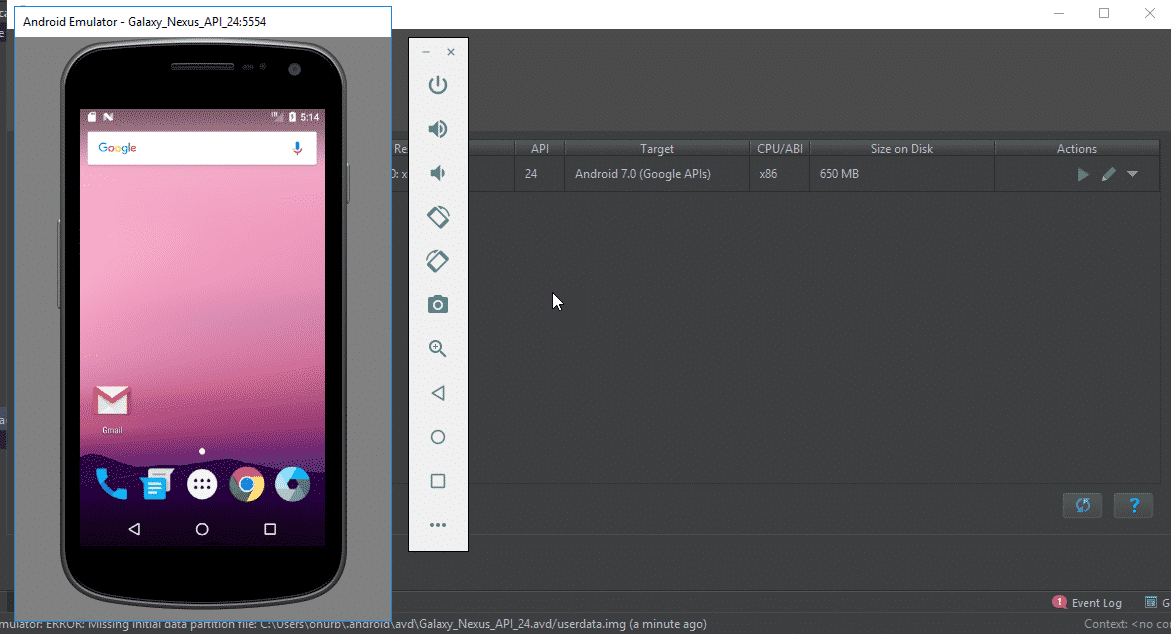
**Step 17)**click "Finish" button



**Step 18)**click the run icon and launch your device.



**Step 19)** You will see the selected AVD on your screen as shown below.

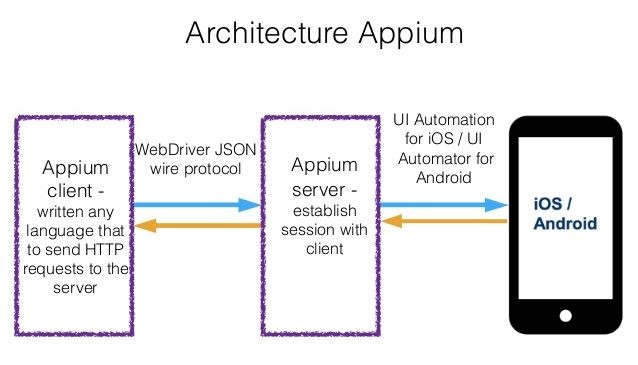


**Step 20)**Add ANDROID\_HOME in Environment Variables. (**I will explain this point in the next article)**

**What is Appium?**

Appium is an open source tool to automate native, web and hybrid mobile application.

Appium supports multiple programming languages like Java, Python, C#, Ruby, Php, Javascript, Robot Framework.

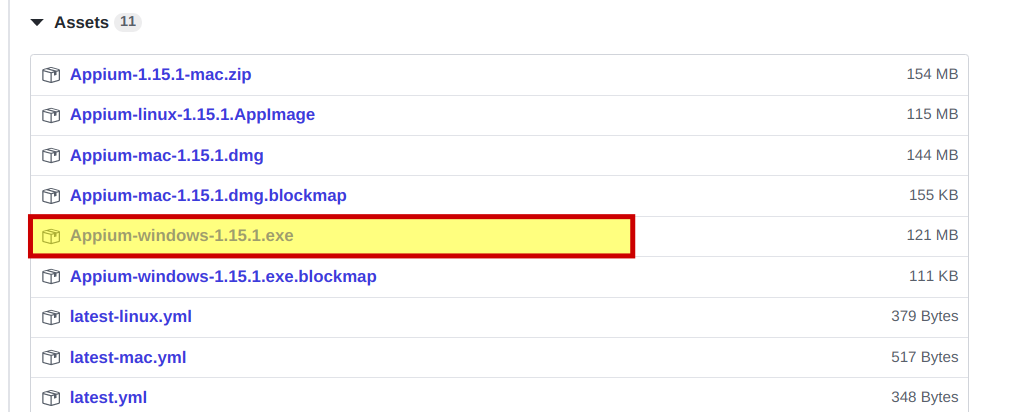


**How to install Appium**

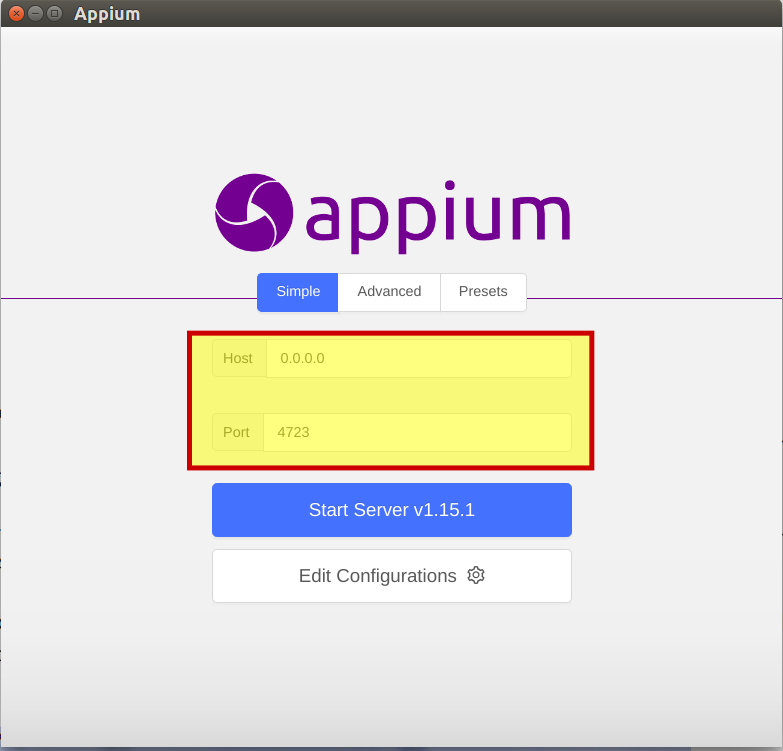
**Step 1)**Go to <http://appium.io/> and click on Download Appium.



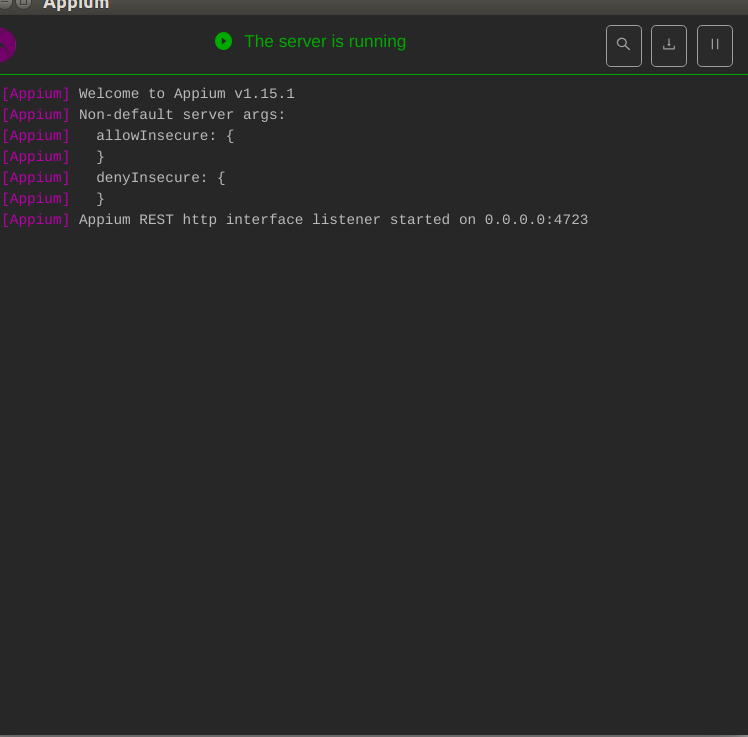
**Step 2)** select the exe file and download.



**Step 3)** Install Appium and open it. It populates the default host and port option which you can change. It also mentions the version of Appium being used.



**Step 4)** On clicking the Start Server Button

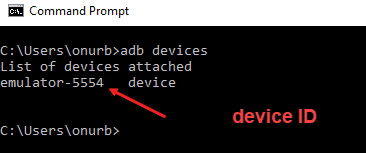


**Android Virtual Device**

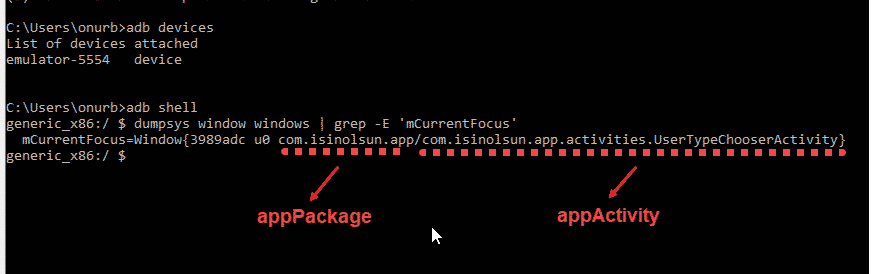
1- Open Android virtual device

2- open calculator application

3- open a command prompt and write “adb devices” command to see connected devices and get the device ID.



4- Write command "adb shell" then write command "dumpsys window windows | grep -E 'mCurrentFocus' " to see the information about app package info and app activity info



**What is Robot Framework?**

* Robot Framework is an open-source test automation framework for acceptance testing and acceptance test-driven development. It follows different test case styles − keyword-driven, behavior-driven and data-driven for writing test cases.
* Robot Framework provides support for external libraries, The most popular library used is AppiumLibrary used for UI testing in a mobile application.

**What should you use Robot Framework?**

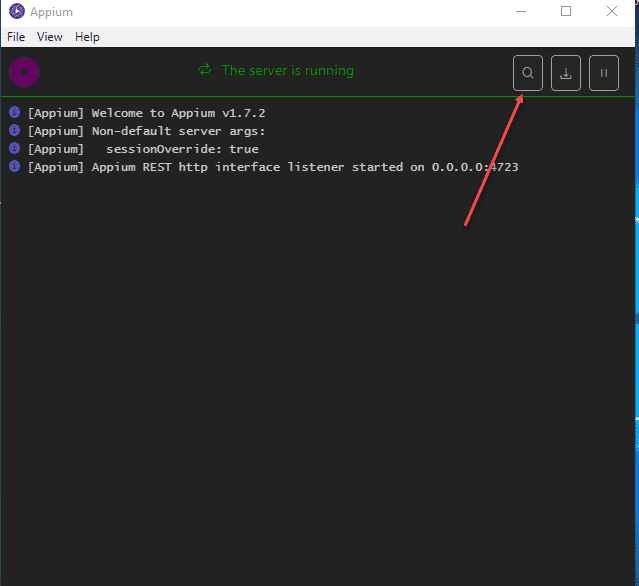
1. Robot framework works fine on all the Operating Systems available.
2. Test cases are written using keyword style in a tabular format. You can use any text editor.
3. It requires little development knowledge to be able to start using its simple and nearly codeless keyword-driven structure to create automated tests with a very short learning curve.
4. This simple structure provides transparency to all the related stakeholders who will be able to easily read and understand the automated tests.
5. Robot test projects can be easily integrated with CI/CD tools such as Jenkins.
6. You can use modular design patterns (such as the popular Page Object Model) to manage your actions and element locators.
7. Provides easy-to-read result reports and logs in HTML format. Supports creating data-driven test cases.
8. Provides tagging to categorize and select test cases to be executed.

**How to install Robot Framework and Appiumlibrary**

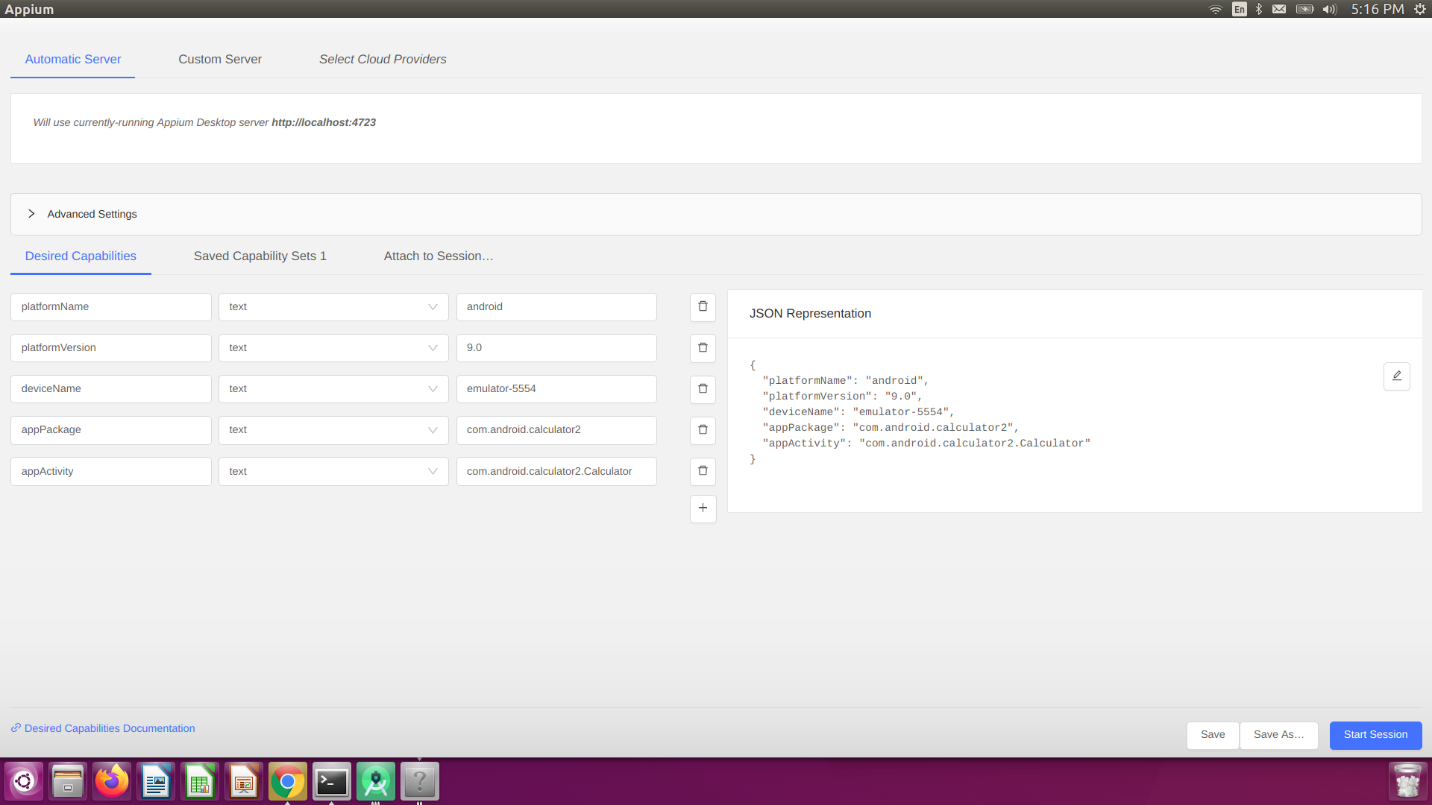
1. open a command prompt and write “python --version” command to check **"Python"** is installed or not on your computer.
2. if Python isn't installed on your computer,[download python and install it.](https://www.python.org/downloads/)
3. Write “pip install robotframework” command to install **Robot Framework**
4. Write "robot --version" to verify installation **Robot Framework**
5. Write "pip install robotframework-appiumlibrary" to install **AppiumLibrary**
6. Install Appium command server
   1. > brew install node # get node.js
   2. > npm install -g appium # get Appium
   3. > npm install -g Appium-doctor # get Appium Doctor to check system ready
   4. > npm install wd # get appium client
   5. > appium & # start appium
   6. > node your-appium-test.js

**Write the first test case using Robot Framework**

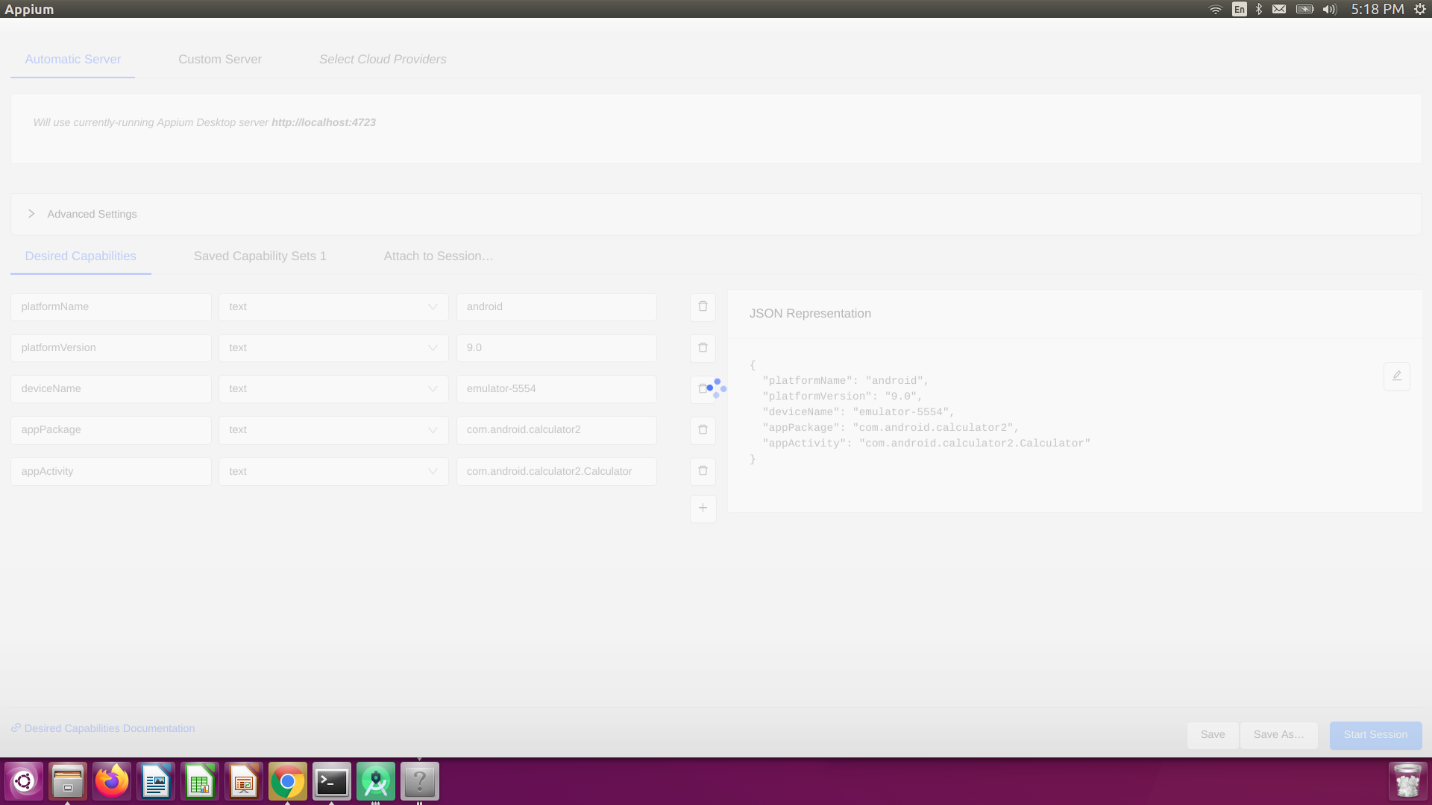
1. go to the **Appium** server and click the magnifier icon to open the inspector.



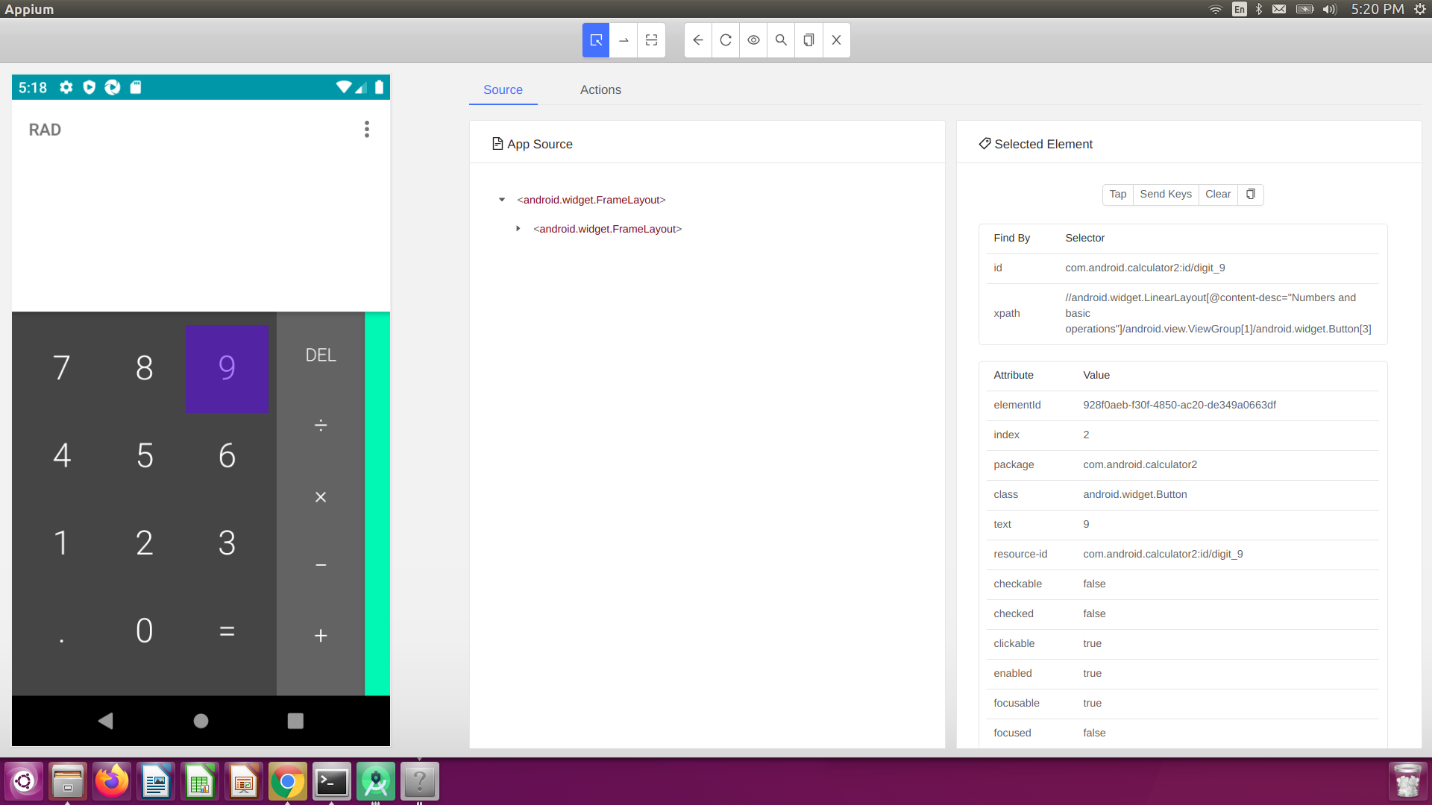
2. Enter the capabilities of your device as shown below in the inspector.



3. Click “Start Session” to start the inspector session to get your mobile elements ids and classes and XPath



4. Get a mobile element’s class or id or XPath



5. write a sample program to open the calculator application and adding two number in the file (test.robot).

\*\*\* Settings \*\*\*

Library    AppiumLibrary

\*\*\* Variables \*\*\*

${REMOTE\_URL}     http://127.0.0.1:4723/wd/hub

${PLATFORM\_NAME}    Android

${PLATFORM\_VERSION}    9.0

${DEVICE\_NAME}    emulator-5554

${Activity\_NAME}        com.android.calculator2.Calculator

${PACKAGE\_NAME}     com.android.calculator2

\*\*\* Keywords \*\*\*

Open calculator

  Open Application   ${REMOTE\_URL}

  ...        platformName=${PLATFORM\_NAME}

  ...    platformVersion=${PLATFORM\_VERSION}

  ...   deviceName=${DEVICE\_NAME}

  ...   automationName=UiAutomator2

    ...    newCommandTimeout=2500

    ...    appActivity=${Activity\_NAME}

    ...    appPackage=${PACKAGE\_NAME}

\*\*\* Test Cases \*\*\*

First Test cases

    Open calculator

    Click Element    xpath=//android.widget.Button[contains(@text,'6')]

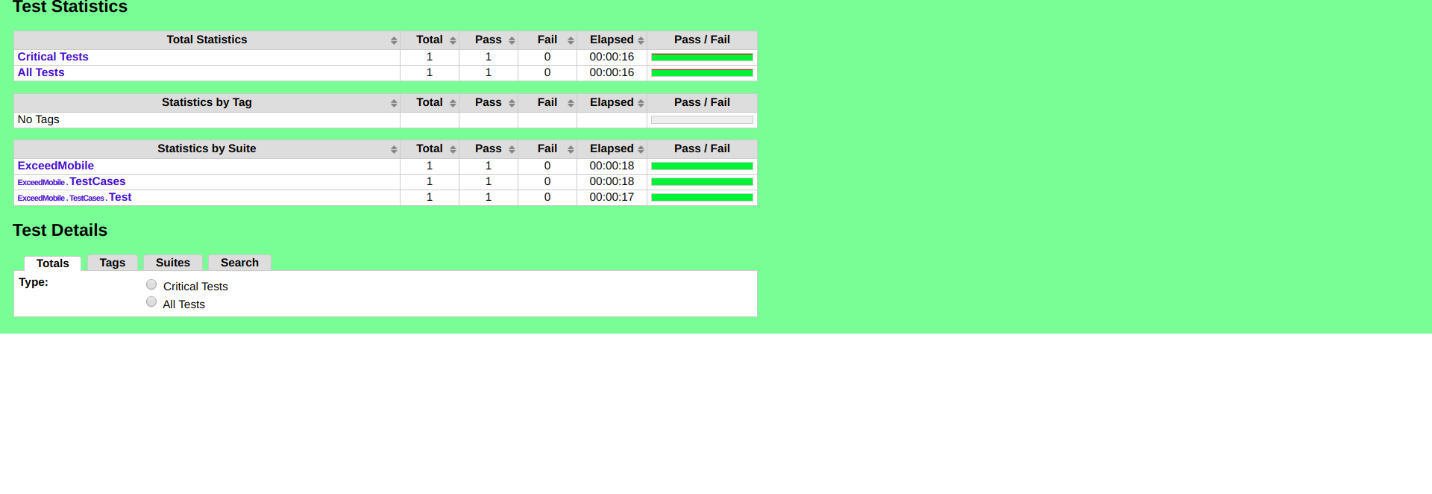
    Click Element    xpath=//android.widget.Button[contains(@text,'+')]

    Click Element    xpath=//android.widget.Button[contains(@text,'4')]

    Click Element    xpath=//android.widget.Button[contains(@text,'=')]

6. Run the program using command "robot test. robot"

7- Robot Framework will generate a report



# How to run Robot Framework test from command line?

[#robotframework](https://dev.to/t/robotframework) [#tutorial](https://dev.to/t/tutorial) [#commandlineinterface](https://dev.to/t/commandlineinterface)

[juperala profile imageJuho Perälä](https://dev.to/juperala)Mar 26, 2020 ・3 min read

Yesterday a colleague learning test automation forgot the syntax for running Robot Framework test cases from command-line. Although full command-line syntax is described on the [Robot Framework User Guide](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html) it is rather long and tedious document to dive into.

This post aims to provide quick introduction for executing single, set or all Robot Framework test cases in a project.

# Basic syntax

The basic syntax for executing Robot Framework tests from command line is:

# basic syntax

robot [options] robot\_files

# example with options

robot -v URL:http://example.com example.robot

For full list of command-line options see robot --help or robot -h option.

# Execute all test cases in folder(s)

To run all robot tests in the folder (including subfolders) use .

# execute all tests in all robot files in current folder and subfolders

robot .

# Execute all test cases in single file

To run all robot tests in single robot file:

# execute all tests in single robot file in current folder

robot example.robot

# execute all tests in single robot file in subfolder

robot path/to/example.robot

# Execute test cases by test name

To run test cases with specific test name use --test or -t option:

# execute test cases with name "Example" in any file.

robot --test Example .

# execute test cases with name "Example" in specific file.

robot --test Example example.robot

Also partial test names and patterns can be used with --test option, see [Using partial names and filter patterns](https://dev.to/juperala/how-to-run-robot-framework-test-from-command-line-5aa#using-partial-names-and-filter-patterns).

# Execute test cases by tags

Test cases and suites annotated with tags (using [Tags] or Force Tags syntax) can be executed by selecting tags to be included or excluded.

Example test suite:

\*\*\* Settings \*\*\*

Force Tags suite

\*\*\* Test Cases \*\*\*

Test One

[Tags] one

No operation

Test Two

[Tags] two

No operation

Also partial tag names and patterns can be used with --include and --exclude option, see [Using partial names and filter patterns](https://dev.to/juperala/how-to-run-robot-framework-test-from-command-line-5aa#using-partial-names-and-filter-patterns).

## Include test cases by tag

To run test cases with specific tag name included use --include or -i option:

# execute test cases with tag "one" in any file.

robot --include one .

# execute test cases with tags "one" and "two" in any file.

robot --include oneANDtwo .

# execute test cases with tag "one" or "two" in any file.

robot --include oneORtwo .

# execute test cases with tag "one" but without tag "two" in any file.

robot --include oneNOTtwo .

## Exclude test cases by tag

To run test cases with specific tag name excluded use --exclude or -e option:

# execute test cases without tag "two" in any file.

robot --exclude two .

# Execute test cases by suite name

In Robot Framework test folders and .robot files are considered as test suites.

Example suite structure:

- +-+- tests

- +-+- feature1

- | +--- positive.robot

- | +--- negative.robot

- +-+- feature2

- +--- positive.robot

- +--- negative.robot

To run test cases in specific test suiteuse --suite or -s option:

# execute test cases from suites named "positive" in any file.

robot --suite positive .

# execute test cases from suite "feature1\positive" in any file.

robot --suite feature1.positive .

Also partial suite names and patterns can be used with --suite option, see [Using partial names and filter patterns](https://dev.to/juperala/how-to-run-robot-framework-test-from-command-line-5aa#using-partial-names-and-filter-patterns).

# Execute failed tests

There is also possibility to rerun all failed test cases and test suites.

## Execute only failed test cases

To rerun failed test cases use --rerunfailed or -R option:

# execute test cases failed in previous run (saved in output.xml)

robot --rerunfailed output.xml .

## Execute failed test suites

To rerun test suites with failed test cases use --rerunfailedsuites or -S option:

# execute test cases with failed test cases in previous run (saved in output.xml)

robot --rerunfailedsuites output.xml .

# Using partial names and filter patterns

The presented --test, --suite, --include and --exclude options also support using partial names and filter patterns to match multiple names and tags:

# execute test cases containing name "Example" in any file.

robot --test \*Example\* .

# execute test cases "Example One" and "Example Two" in any file.

robot --test "Example [One|Two]" .

# execute test cases with tags starting with "One" in any file.

robot --include One\* .

# execute test cases without tags ending with "Two" in any file.

robot --exclude \*Two .

# execute test cases from suites starting with "positive" in any file.

robot --suite positive\* .

For full list of supported filter patterns see User Guide section [Simple Patterns](https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#simple-patterns)

# Combining filters

The presented --test, --suite, --include and --exclude options can be used also in combination:

# execute test cases containing name "Example" and having tag "One" in any file.

robot --include One --test \*Example\* .

# execute test cases from suite "FeatureA" exluding tests with tag "Smoke" in any file.

robot --suite FeatureA --exclude Smoke .

# execute test cases with tag "Pending" from specific file.

robot --exclude Pending example.robot