

Automated Testing with Appium



Automated Testing



<http://appium.io/docs/en/about-appium/intro/>



Automated Testing

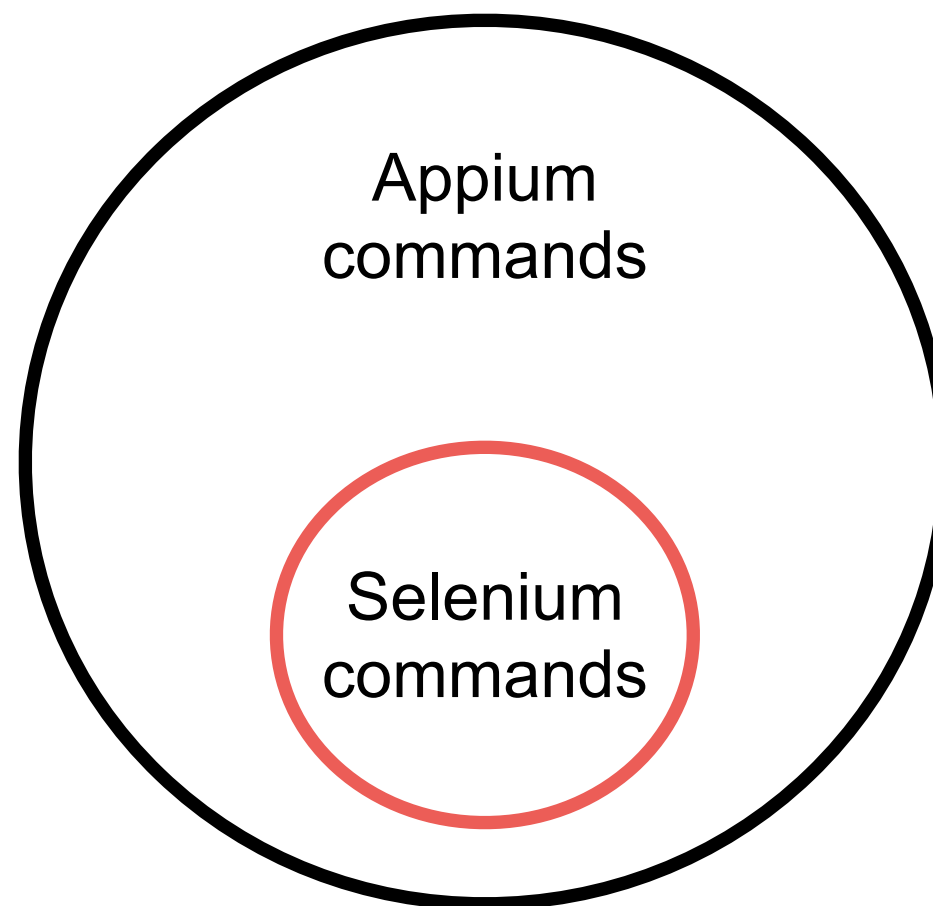


<https://appium.github.io/appium/docs/en/2.0/>

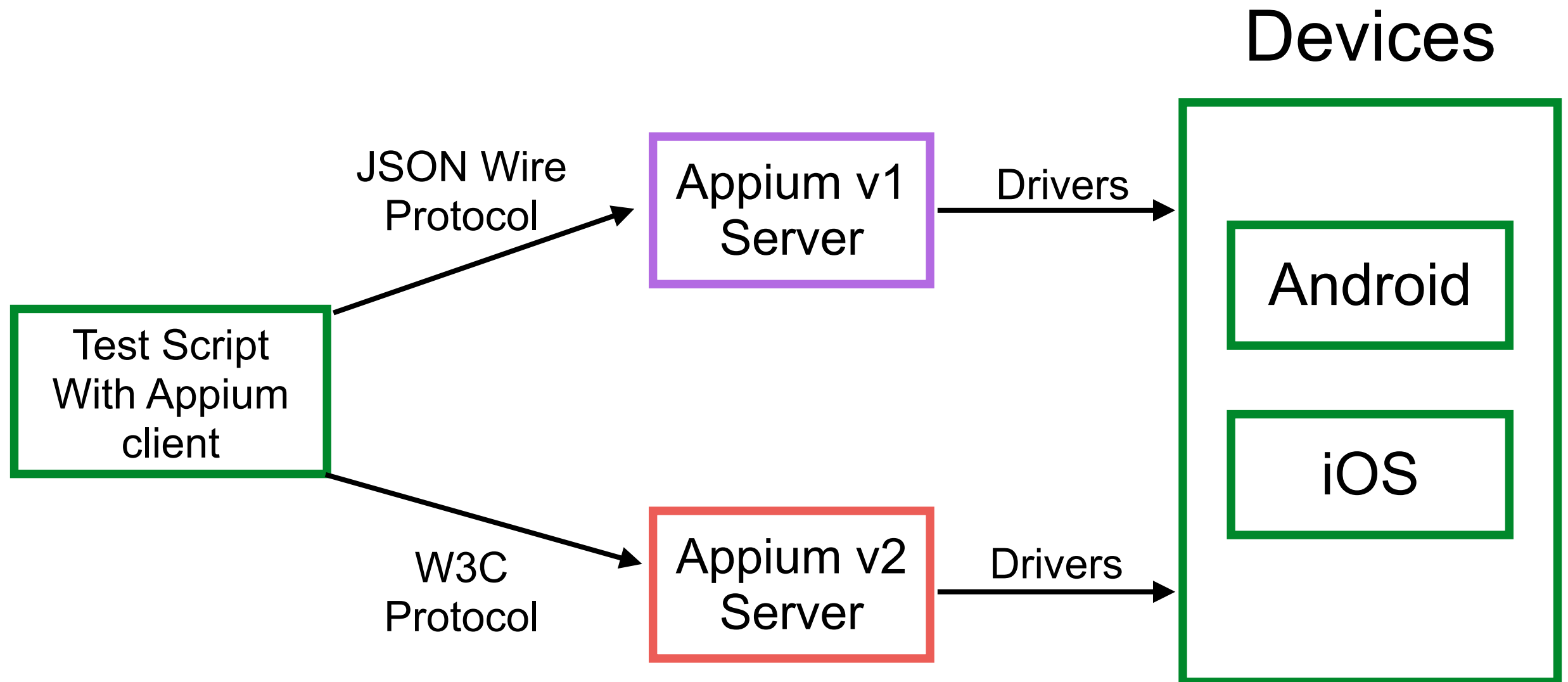


Appium

Automation tool for mobile app
Selenium for mobile
High compatibility with selenium



Architecture



Appium Tools

Appium Server via npm

Appium Server GUI/Desktop

Appium Inspector

<http://appium.io/docs/en/about-appium/getting-started/?lang=en#installing-appium>



Appium Clients

Ruby
Python
Java
C#

<http://appium.io/docs/en/about-appium/appium-clients/index.html>

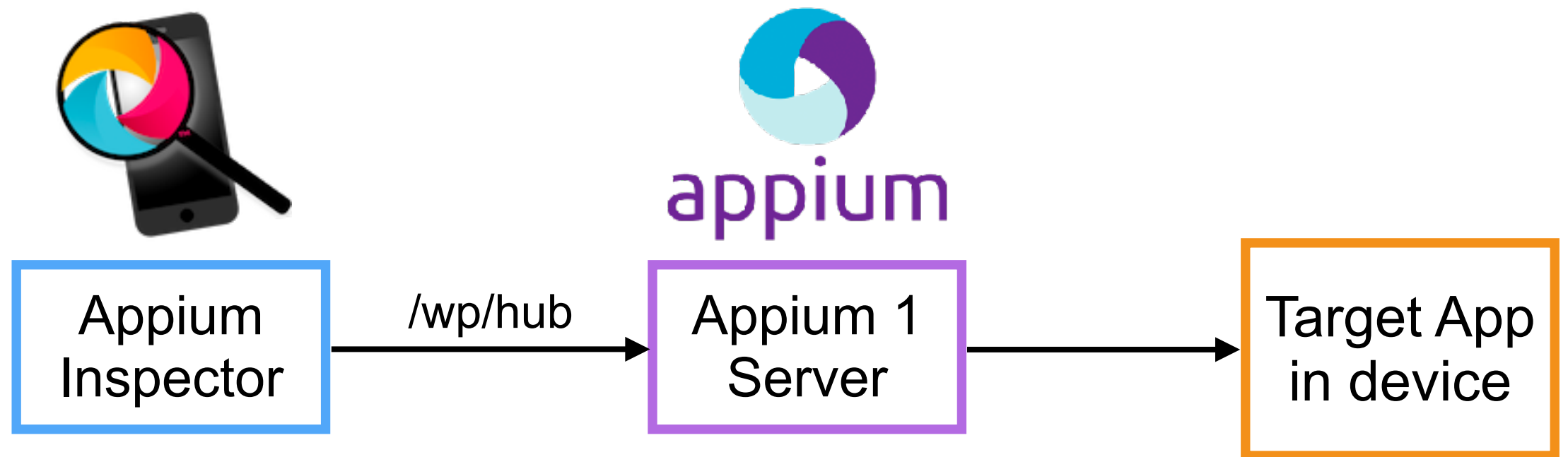


Appium Drivers

Platform	Driver	Platform Versions	Appium Version
iOS	XCUITest	9.3+	1.6.0+
	UIAutomation	8.0 to 9.3	All
Android	Espresso	?+	1.9.0+
	UiAutomator2	?+	1.6.0+
	UiAutomator	4.3+	All
Mac	Mac	?+	1.6.4+
Windows	Windows	10+	1.6.0+



Let's start with Appium



Steps to run

Appium doctor

Start server

Provide target apps (api, ipa/app)

Appium Inspector



Appium Docker

```
$npm install -g appium-doctor  
$appium-doctor
```

```
$npm install -g @appium/doctor
```

<https://github.com/appium/appium-doctor>



Appium Docker

\$appium-doctor

```
info AppiumDoctor Appium Doctor v.1.12.1
info AppiumDoctor ### Diagnostic for necessary dependencies starting ###
info AppiumDoctor ✓ The Node.js binary was found at: /Users/somkiat/.volta/tools/image/node
info AppiumDoctor ✓ Node version is 14.20.0
info AppiumDoctor ✓ Xcode is installed at: /Applications/Xcode.app/Contents/Developer
info AppiumDoctor ✓ Xcode Command Line Tools are installed in: /Applications/Xcode.app/Contents/Developer
info AppiumDoctor ✓ DevToolsSecurity is enabled.
info AppiumDoctor ✓ The Authorization DB is set up properly.
WARN AppiumDoctor ✗ Carthage was NOT found!
info AppiumDoctor ✓ HOME is set to: /Users/somkiat
WARN AppiumDoctor ✗ ANDROID_HOME is NOT set!
WARN AppiumDoctor ✗ JAVA_HOME is NOT set!
WARN AppiumDoctor ✗ adb could not be found because ANDROID_HOME is NOT set!
WARN AppiumDoctor ✗ android could not be found because ANDROID_HOME is NOT set!
WARN AppiumDoctor ✗ emulator could not be found because ANDROID_HOME is NOT set!
WARN AppiumDoctor ✗ Bin directory for $JAVA_HOME is not set
info AppiumDoctor ### Diagnostic for necessary dependencies completed, 7 fixes needed. ###
info AppiumDoctor ### Diagnostic for optional dependencies starting ###
WARN AppiumDoctor ✗ opencv4nodejs cannot be found.
info AppiumDoctor ✓ ffmpeg is installed at: /usr/local/bin/ffmpeg. ffmpeg version 5.1 Copy
the FFmpeg developers
WARN AppiumDoctor ✗ mjpeg-consumer cannot be found.
WARN AppiumDoctor ✗ idb and idb_companion are not installed
info AppiumDoctor ✓ applesimutils is installed at: /usr/local/bin/applesimutils. Installed
```



Appium Server

Install via npm

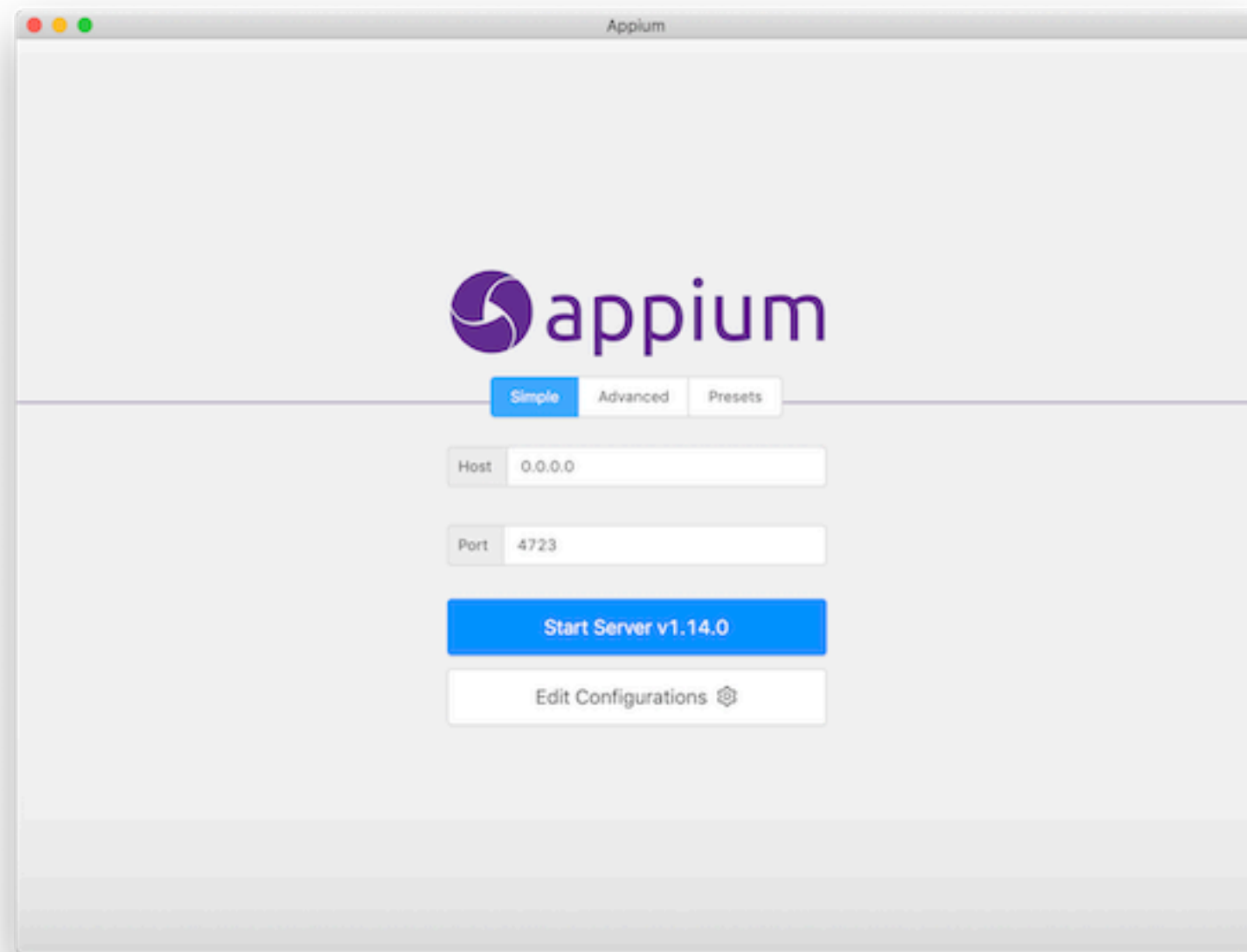
```
$npm install -g appium  
$appium
```

<https://github.com/appium/appium>



Appium Server

Use GUI tool => Appium Desktop

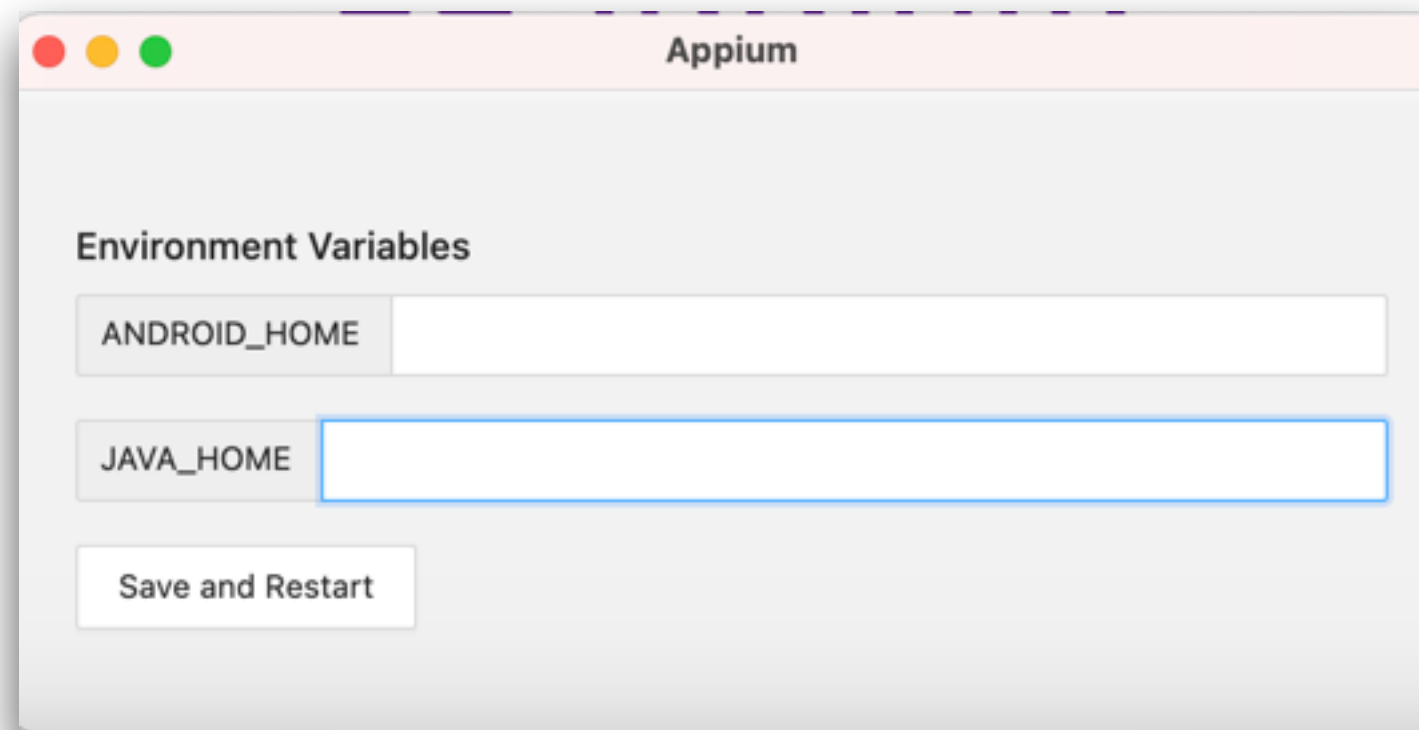


<https://github.com/appium/appium-desktop>



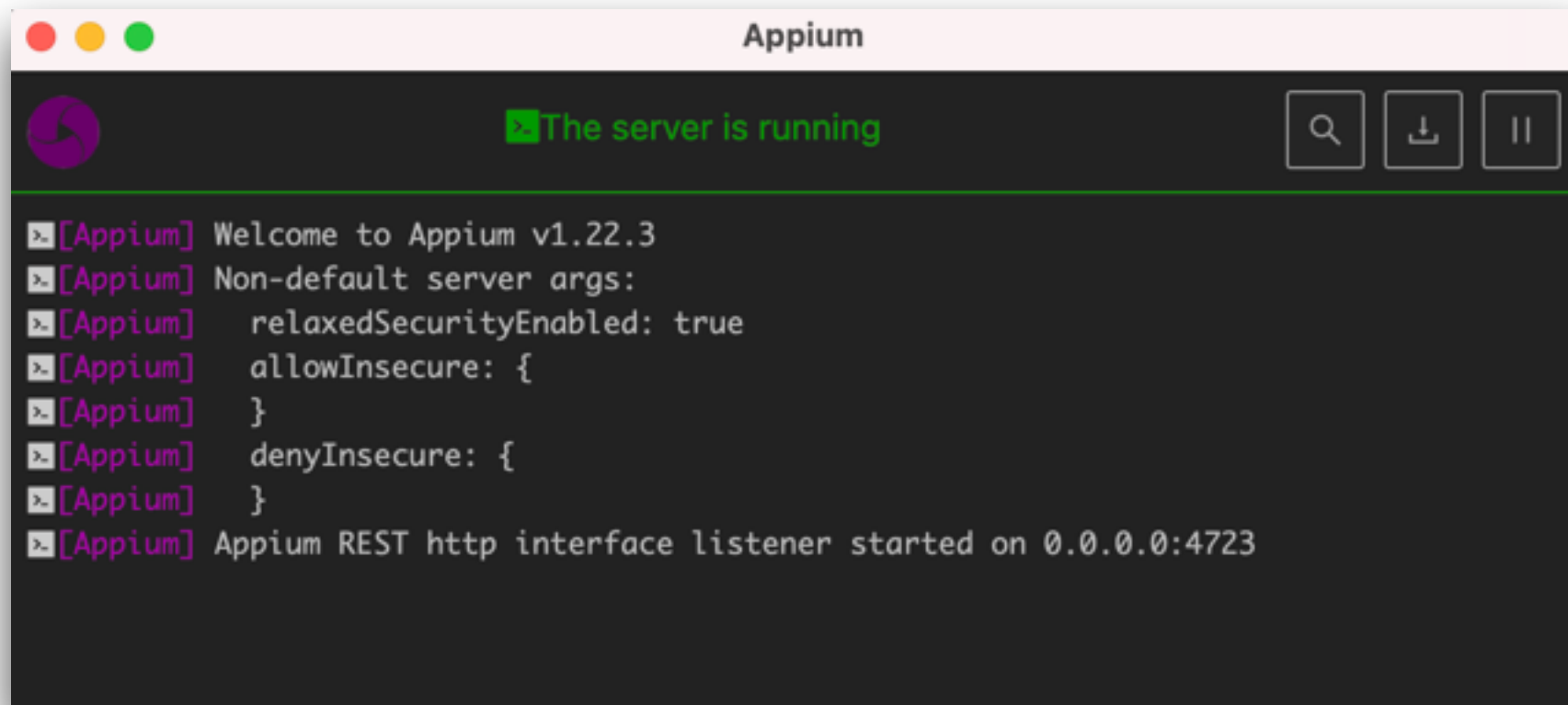
Appium Server

Configurations of server



Appium Server

Start server



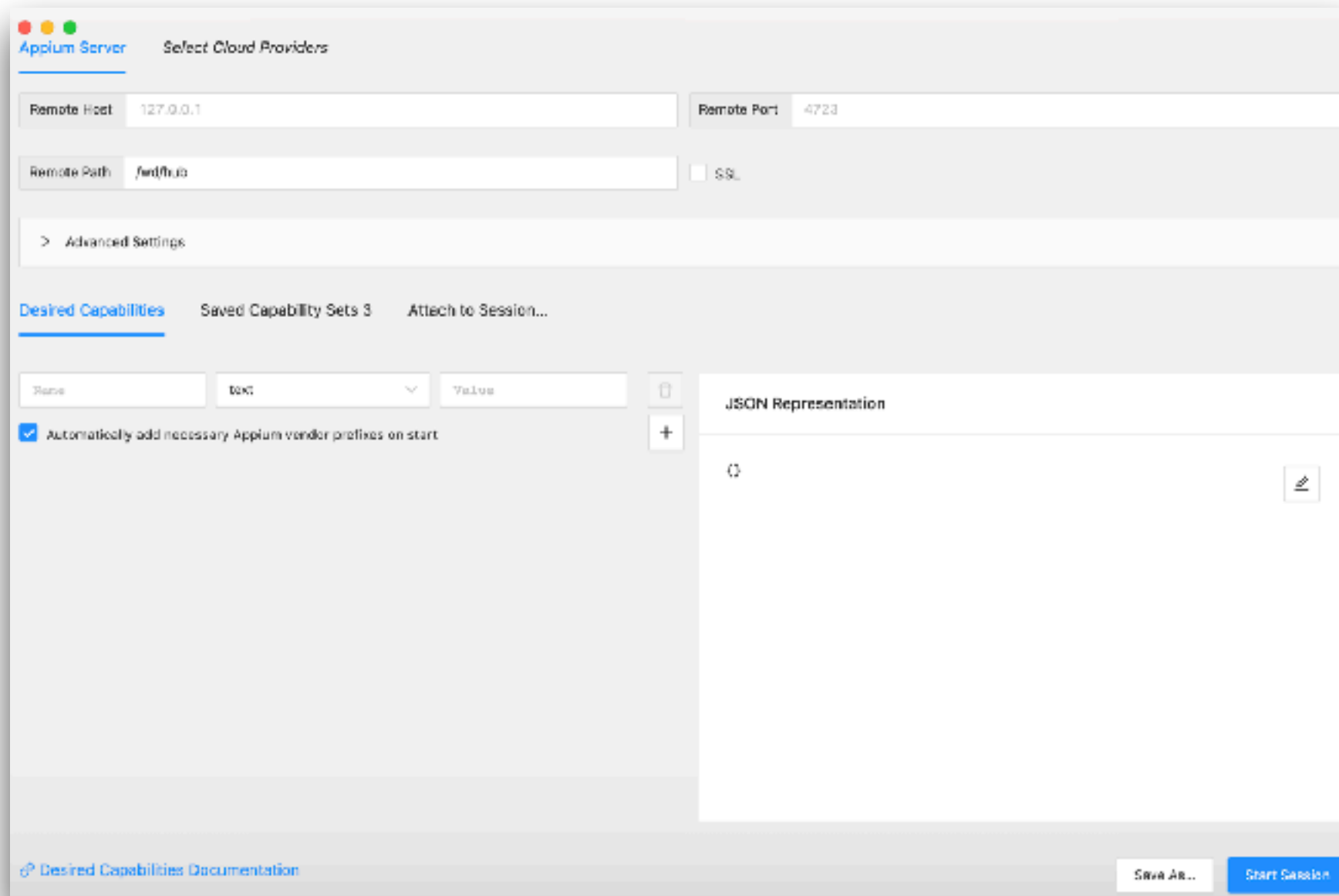
The image shows a screenshot of the Appium application window. The title bar is pink and says "Appium". The main area has a dark background. At the top, there's a status bar with a green checkmark icon and the text "The server is running". To the right of this are three icons: a magnifying glass, a download arrow, and a pause symbol. Below the status bar, there's a list of log messages, each preceded by a small icon and the text "[Appium]". The logs show the welcome message, non-default server arguments (relaxedSecurityEnabled: true, allowInsecure: {}, denyInsecure: {}), and the REST http interface listener starting on 0.0.0.0:4723.

```
[Appium] Welcome to Appium v1.22.3
[Appium] Non-default server args:
[Appium]   relaxedSecurityEnabled: true
[Appium]   allowInsecure: {
[Appium]   }
[Appium]   denyInsecure: {
[Appium]   }
[Appium] Appium REST http interface listener started on 0.0.0.0:4723
```



Appium Inspector

GUI inspector for mobile apps



<https://github.com/appium/appium-inspector>



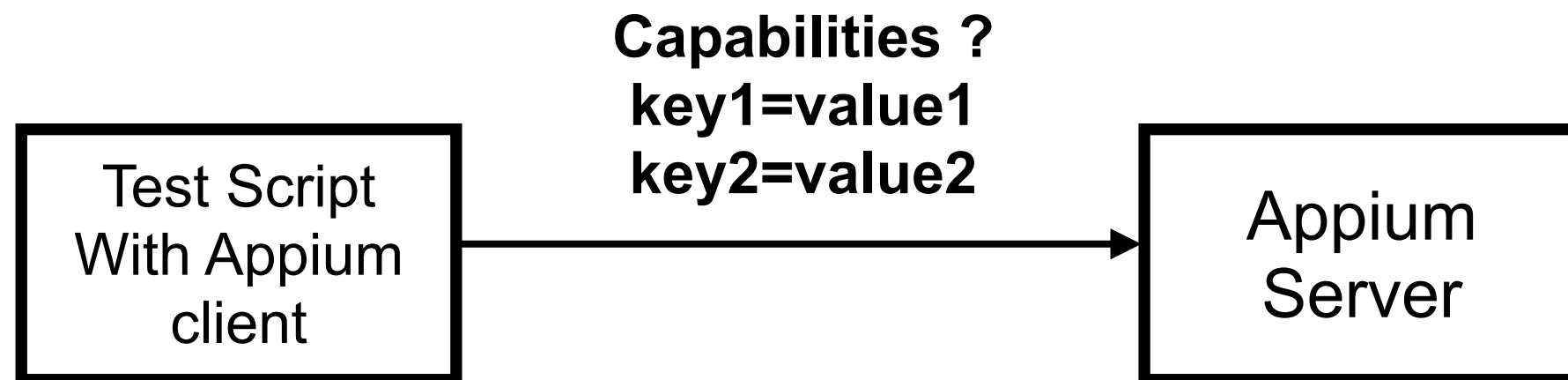
Create a session

HTTP Method	Route	Action
POST	/session	Start an session of test
POST	/session/:sessionId/element	Find an element
POST	/session/:sessionId/ element/:elementId/click	Click/tab on selected element

<http://appium.io/docs/en/commands/session/create/>



Create a session with capabilities



<http://appium.io/docs/en/writing-running-appium/caps/index.html>



Required capabilities

Name	Description
platformName	Platform to automate (iOS, Android)
platformVersion	Version of platform
deviceName	Type of device to automate
app	Path to your app



Session capabilities for android

```
{  
  "platformName": "Android",  
  "automationName": "UiAutomator2",  
  "app": "Path to APK file",  
  "deviceName": "ID/Name of target device" ?  
}
```

<https://appium.io/docs/en/drivers/android-uiautomator2/>

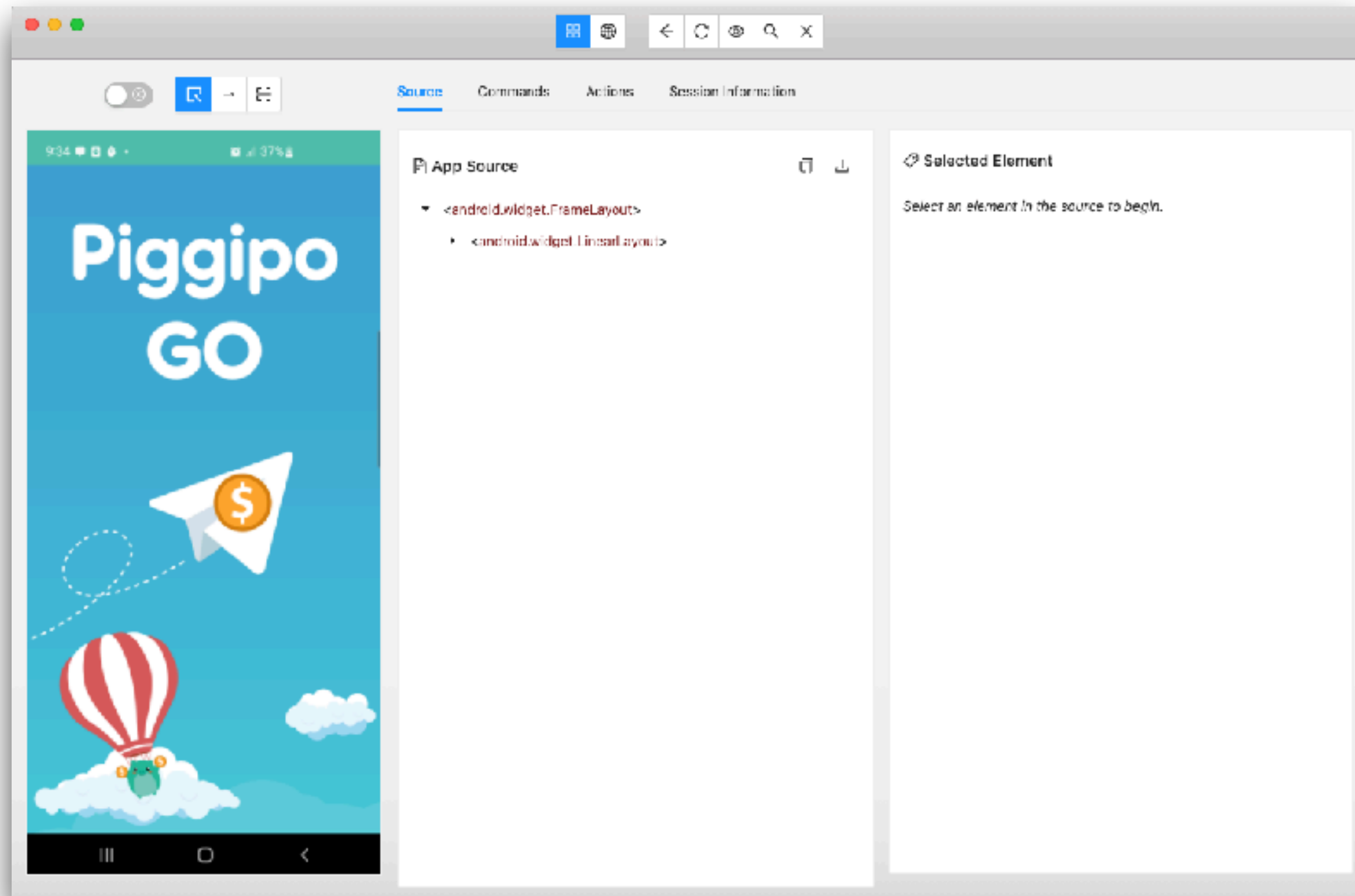


List of android's devices

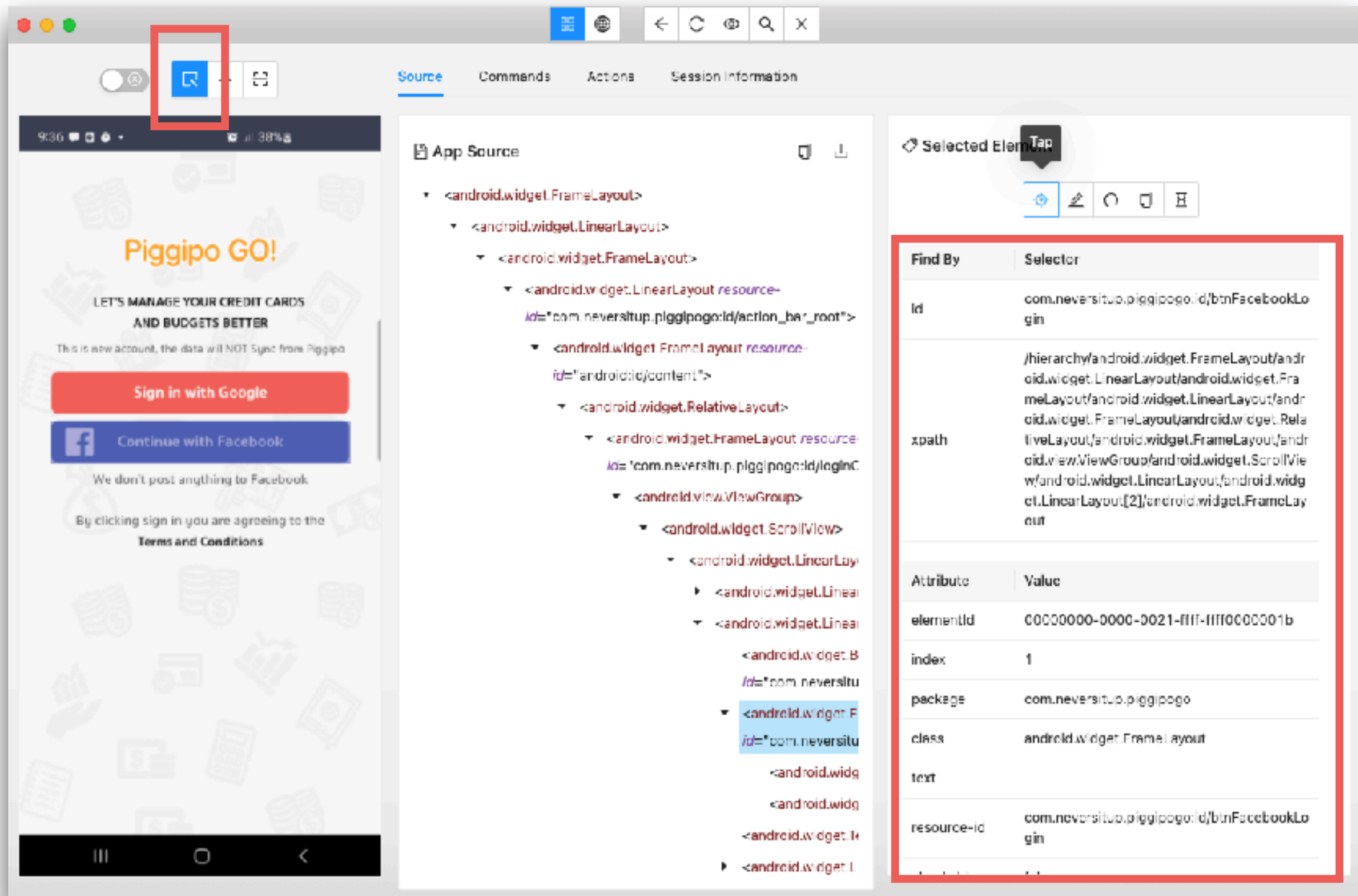
\$adb devices



Appium Inspector



Inspect Element



Simulate actions on element

The screenshot displays the Android Studio environment. On the left, a mobile app is running in a virtual device, showing a login screen for 'Piggipo GO!'. The screen includes a status bar at the top with the time 9:36 and battery level 38%. The app content features the text 'Piggipo GO!', 'LET'S MANAGE YOUR CREDIT CARDS AND BUDGETS BETTER', a note about new accounts, and two login buttons: 'Sign in with Google' and 'Continue with Facebook'. Below these is a disclaimer about Facebook data and a link to 'Terms and Conditions'.

In the center, the 'App Source' view shows the XML hierarchy of the app's layout. The selected element is a `<android.widget.FrameLayout>` with the attribute `id="com.neversitup.piggipogoid/btnFacebookLogin"`.

On the right, the 'Selected Element' panel is highlighted with a red box. It shows a 'Tap' action being simulated on the selected element. Below this, there are two tables: 'Find By' and 'Attribute'.

Find By	Selector
id	com.neversitup.piggipogoid/btnFacebookLogin
xpath	/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.widget.RelativeLayout/android.widget.FrameLayout/android.view.ViewGroup/android.widget.ScrolledView/android.widget.LinearLayout/android.widget.FrameLayout

Attribute	Value
elementId	00000000-0000-0021-ffff-ffff0000001b
index	1
package	com.neversitup.piggipogo
class	android.widget.FrameLayout
text	
resource-id	com.neversitup.piggipogoid/btnFacebookLogin



Code Example

The screenshot shows the Selenium IDE interface. On the left, a mobile app preview for 'Piggipo GO!' is visible. The main panel displays 'Session Information' for a session titled 'Piggipo GO!'. The session details include a length of 00:02:13, platform LINUX, webStorageEnabled false, and takesScreenshot true. The currently active app ID is com.newersilup.piggipago.

Below the session information, there is a section titled 'Start this Kind of Session with Code'. It provides a sample code snippet for starting a session using the Appium robot client. The code includes settings for the library, test teardown, and suite teardown, as well as variables for the URL, platform, app, device name, automation name, and whether to ensure webviews have pages and take native web screenshots.

```
# This sample code uses the Appium robot client
# pip install robotframework-appiumlibrary
# Then you can paste this into a file and simply run with robot
#
# more keywords on: http://sevhathalan.github.io/robotframework-appiumlibrary/Appium

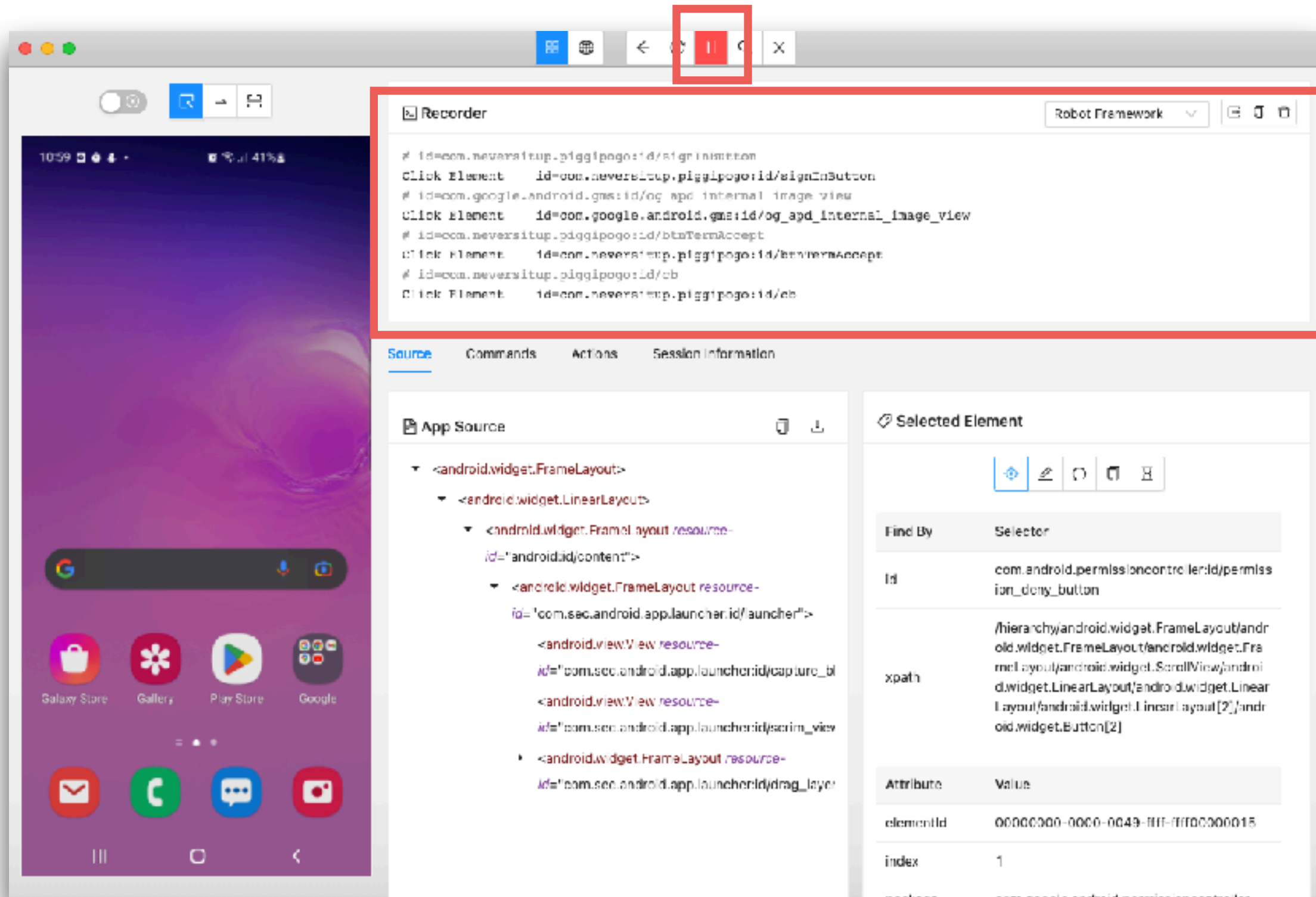
*** Settings ***
Library           AppiumLibrary
Test Teardown     Quit Application
Suite Teardown    Close Application

*** Variables ***
${REMOTE_URL}     http://127.0.0.1:4723/wd/hub
${platformname}   Android
${appium:app}      /Users/sonkian/data/atide/appium/workshop/PiggipoGO_1.2.21.apk
${appium:deviceName}  R55M16QK5JX
${appium:automationName}  UIAutomator2
${appium:ensureWebviewsHavePages}  True
${appium:nativeWebScreenshot}  True
```

A dropdown menu is open, showing the following options: JS - WD (Promise), JS - Webdriver.io, JS - Oxygen HQ, Java - JUnit, Python, Ruby, and Robot Framework (selected).



Try to record test script



Session capabilities for iOS

```
{  
  "platformName": "Android",  
  "automationName": "XCUITest",  
  "app": "Path to IPA file",  
  "platformVersion": "14.5"  
  "deviceName": "ID/Name of target device"  
}
```

<https://appium.io/docs/en/drivers/ios-xcuitest/index.html>



List of iOS's devices

\$xcrun xctrace list devices

```
== Simulators ==
Apple TV Simulator (15.2) (9666637B-AD71-47CD-932D-DE7BA9096F46)
Apple TV 4K (2nd generation) Simulator (15.2) (2B6A15B3-4A03-41B5-B6C6-D7727DCA94D8)
Apple TV 4K (at 1080p) (2nd generation) Simulator (15.2) (EC5B5310-9DE7-4CB2-8AE2-172FC885DB8C)
iPad (9th generation) Simulator (15.2) (03B84395-1A2B-4759-B01A-171CD4F8F796)
iPad Air (4th generation) Simulator (15.2) (6389A3CA-7805-452D-9110-5B3796A9D9BB)
iPad Pro (11-inch) (3rd generation) Simulator (15.2) (40E522A5-14CE-453C-A54C-29D096F19236)
iPad Pro (12.9-inch) (5th generation) Simulator (15.2) (1E6E63EF-FA56-4E31-8082-13B0C664AD33)
iPad Pro (9.7-inch) Simulator (15.2) (9C7181D9-73D6-48BF-8972-80A73ADB2EA9)
iPad mini (6th generation) Simulator (15.2) (01F8F0AB-1AF5-4FBE-9587-B7713F7BBC6F)
iPhone 11 Simulator (15.2) (1E75E325-57EC-4D6D-85BF-9958B0A9B158)
iPhone 11 Pro Simulator (15.2) (7D50F81C-3521-447F-A79C-8613C6C58FEF)
iPhone 11 Pro Max Simulator (15.2) (FB336CEE-A241-43D1-8704-B0632EA9FD28)
iPhone 12 Simulator (15.2) (5325CF3D-9576-462A-A110-49A5D76665FE)
```



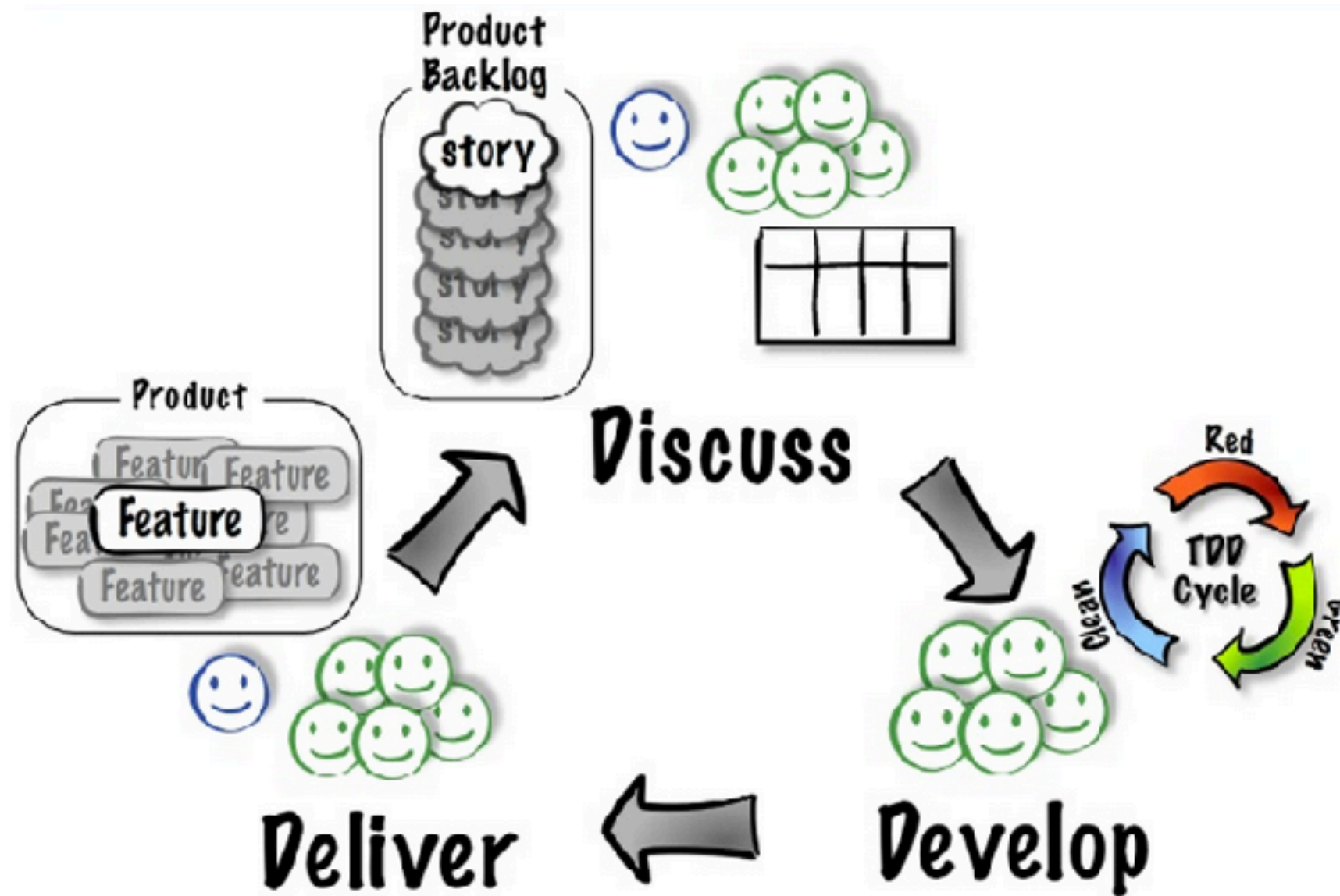
Write your tests



THINK before coding



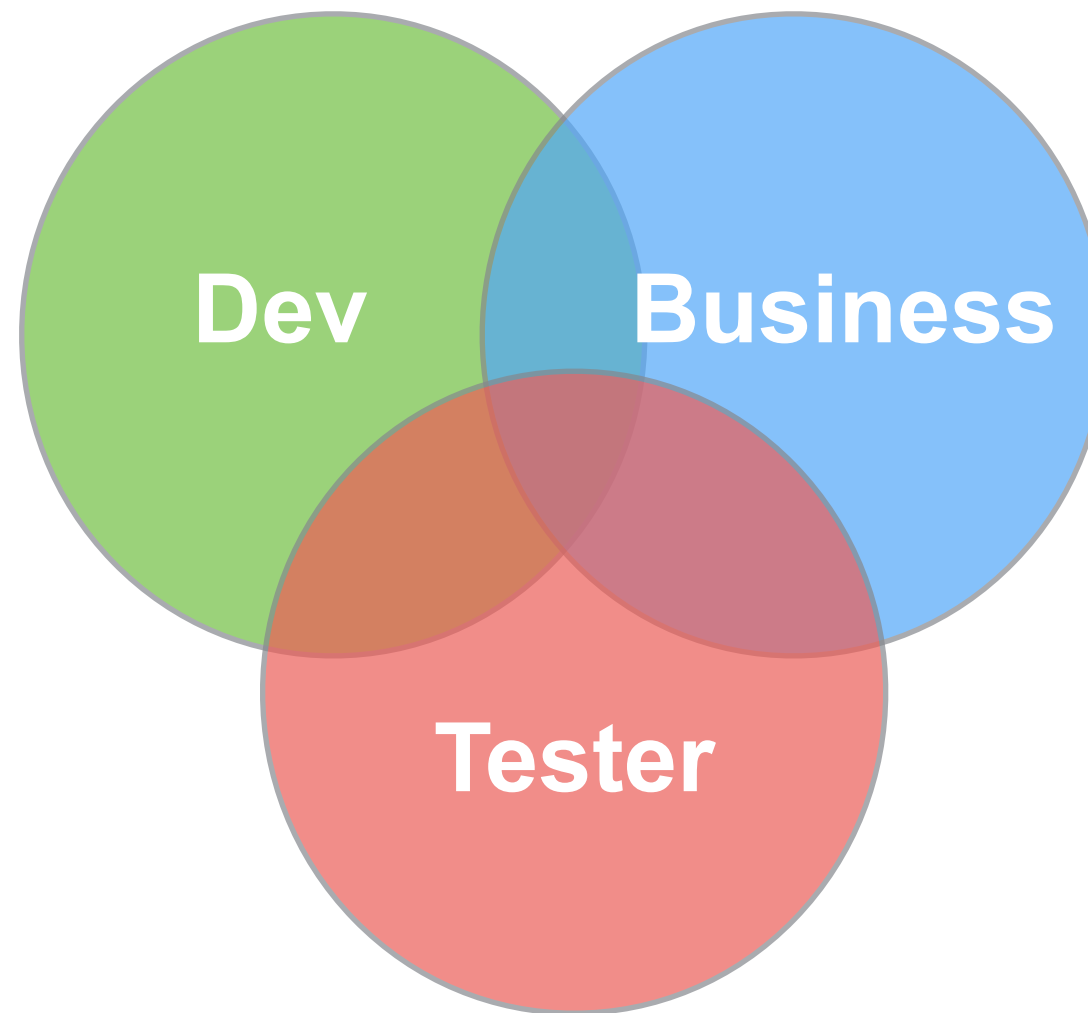
Acceptance Test-Driven Development



(Model developed with Pekka Klärck, Bas Vodde, and Craig Larman.)



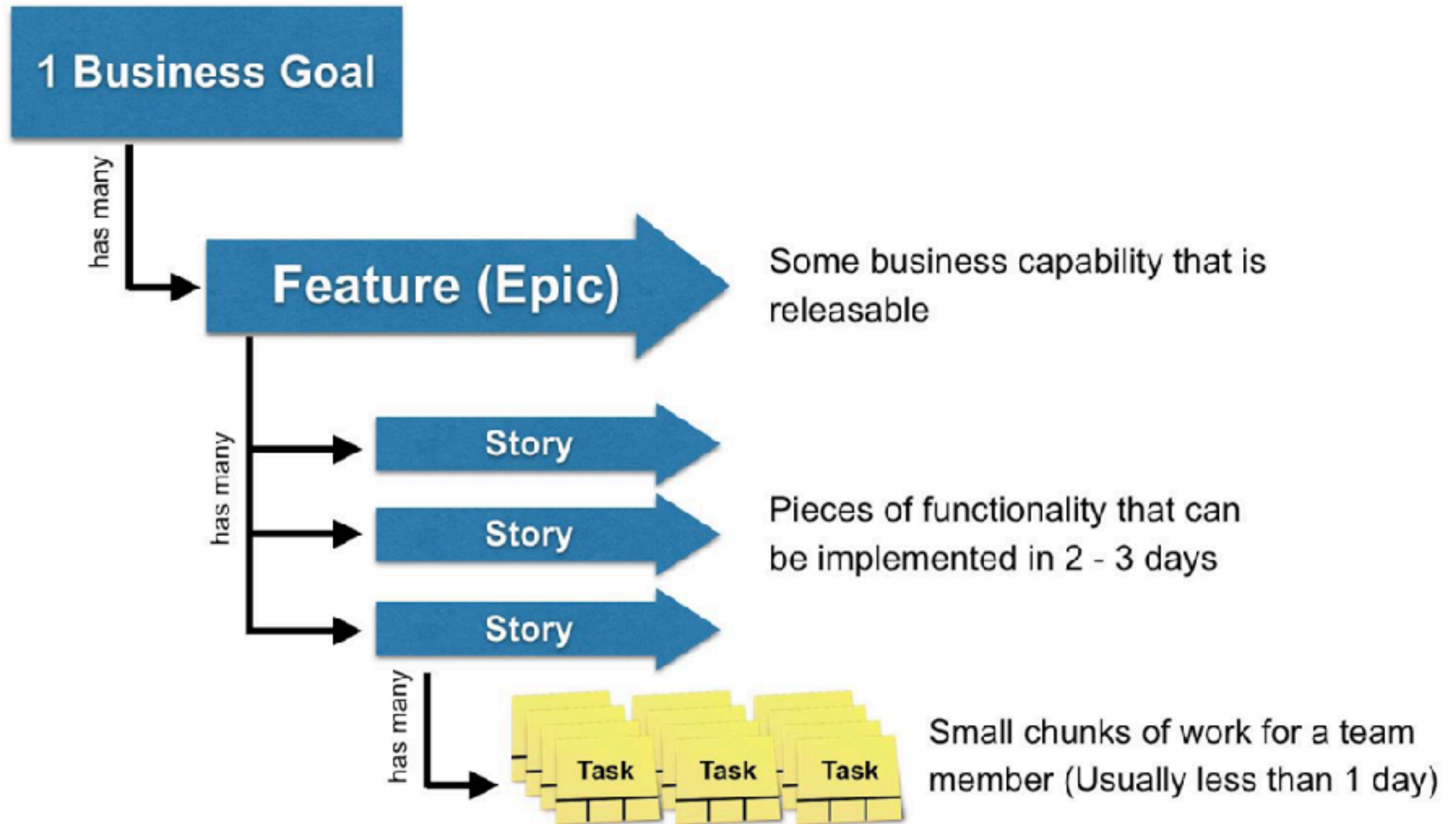
Acceptance Test-Driven Development



Acceptance Tests
=
Business Criteria
+
Examples (data)



Work break down



Iterative and incremental process

Feature 1

Time



Iterative and incremental process

Done = coded and tested



The diagram illustrates an iterative and incremental process. It features a horizontal black arrow pointing to the right, representing the progression of time. Above the arrow, on the left side, is a green rectangular box containing the text "Feature 1". Below the arrow, centered, is the word "Time".

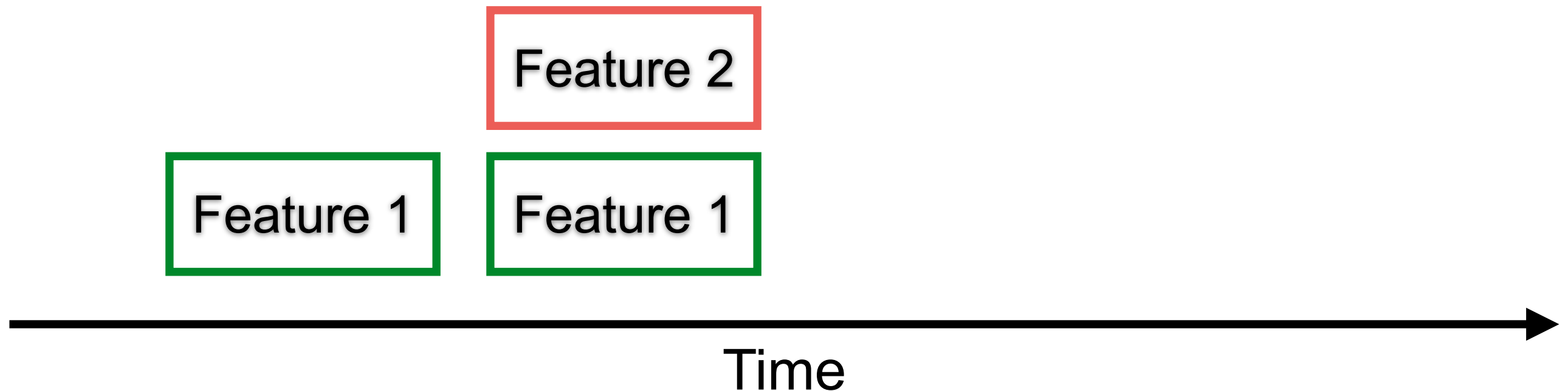
Feature 1

Time



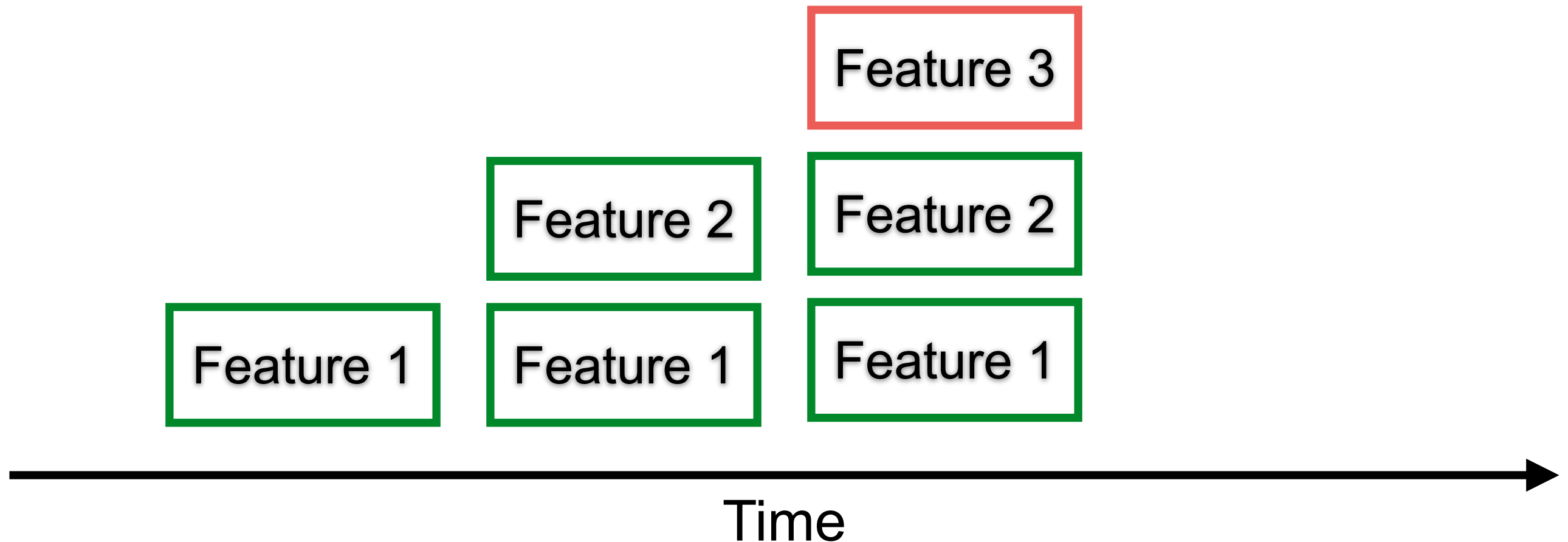
Iterative and incremental process

Done = coded and tested



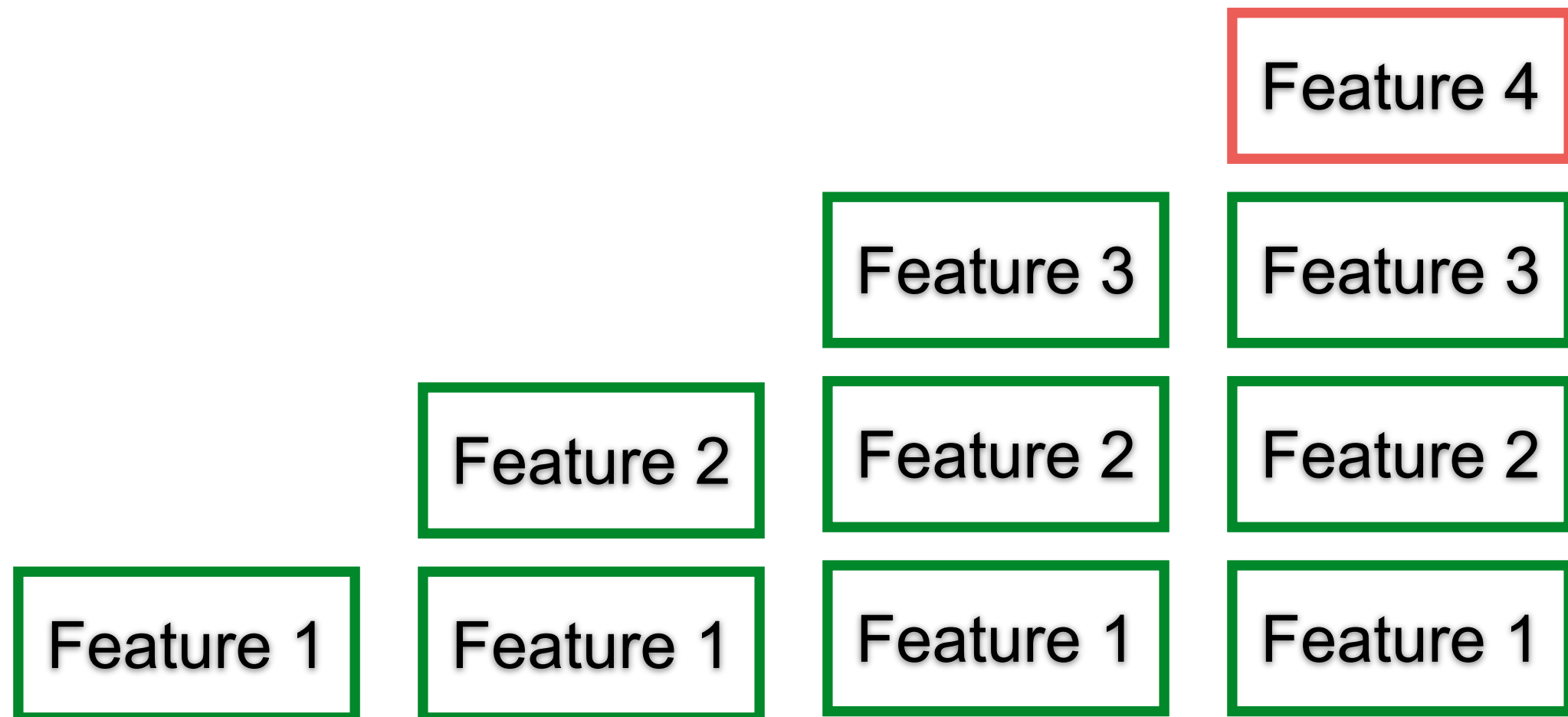
Iterative and incremental process

Done = coded and tested



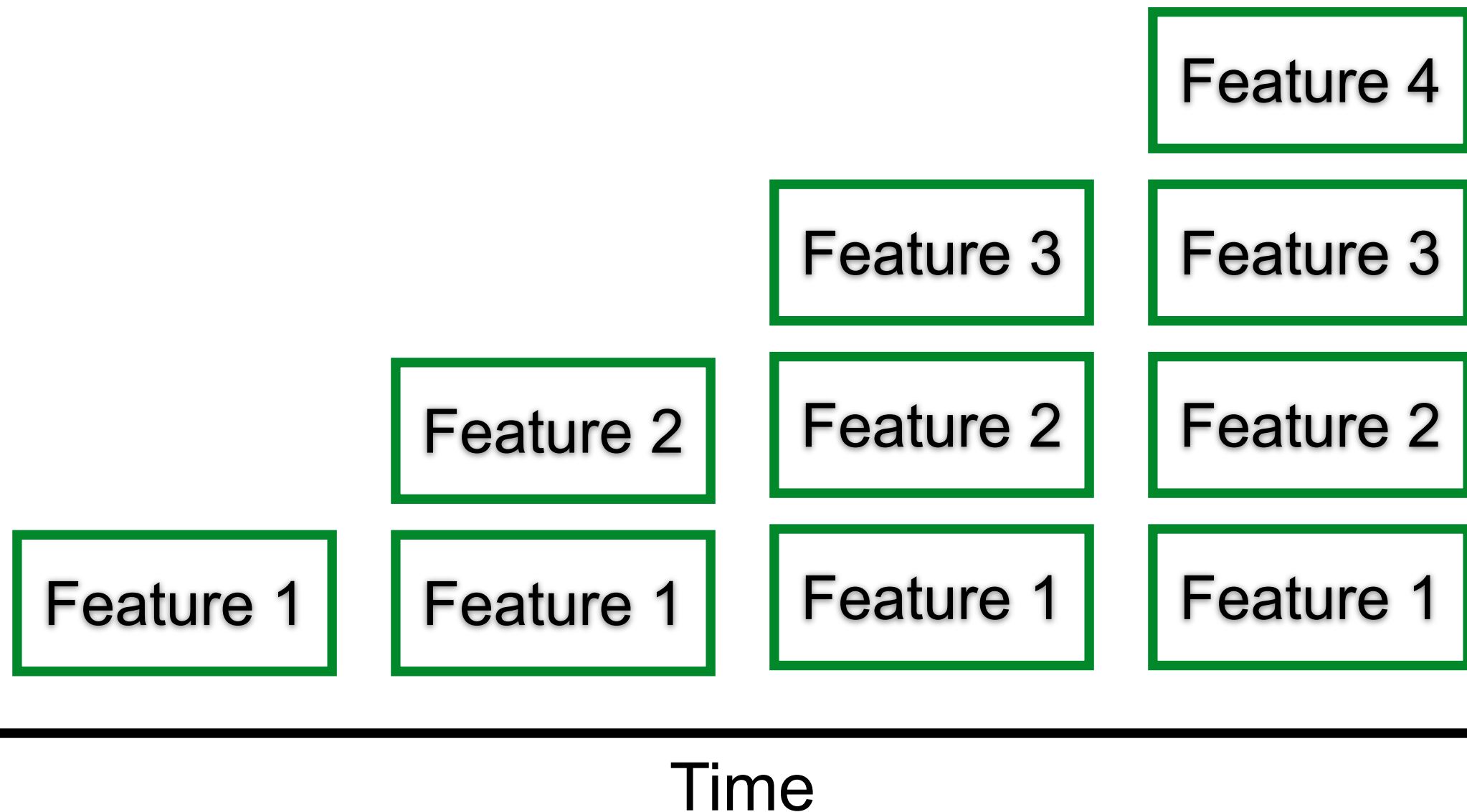
Iterative and incremental process

Done = coded and tested



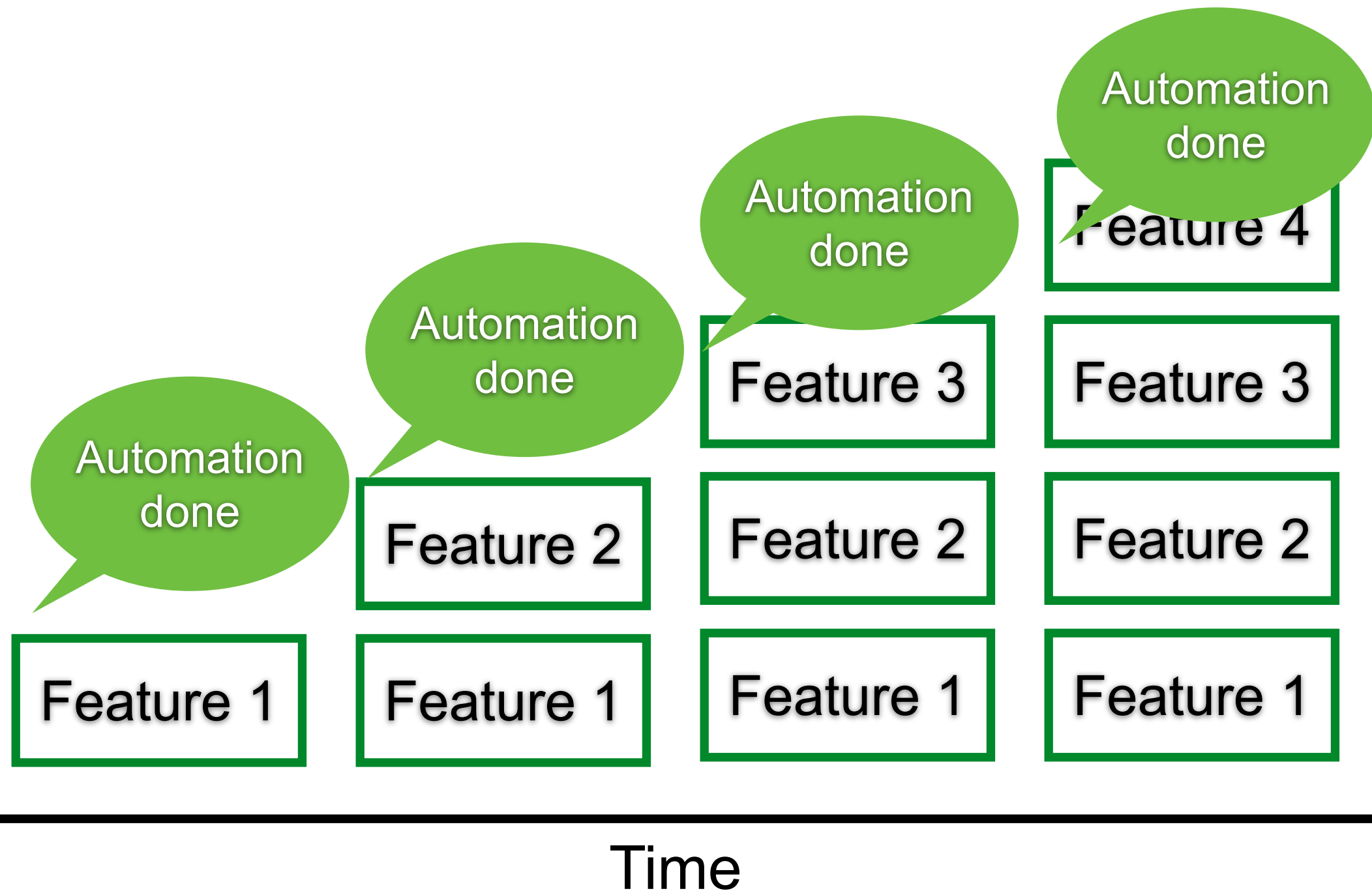
Iterative and incremental process

Done = coded and tested



Iterative and incremental process

Done = coded and tested



Automation feedback

Easier over time ?

Time spent on maintenance ?

Test find regression bugs ?



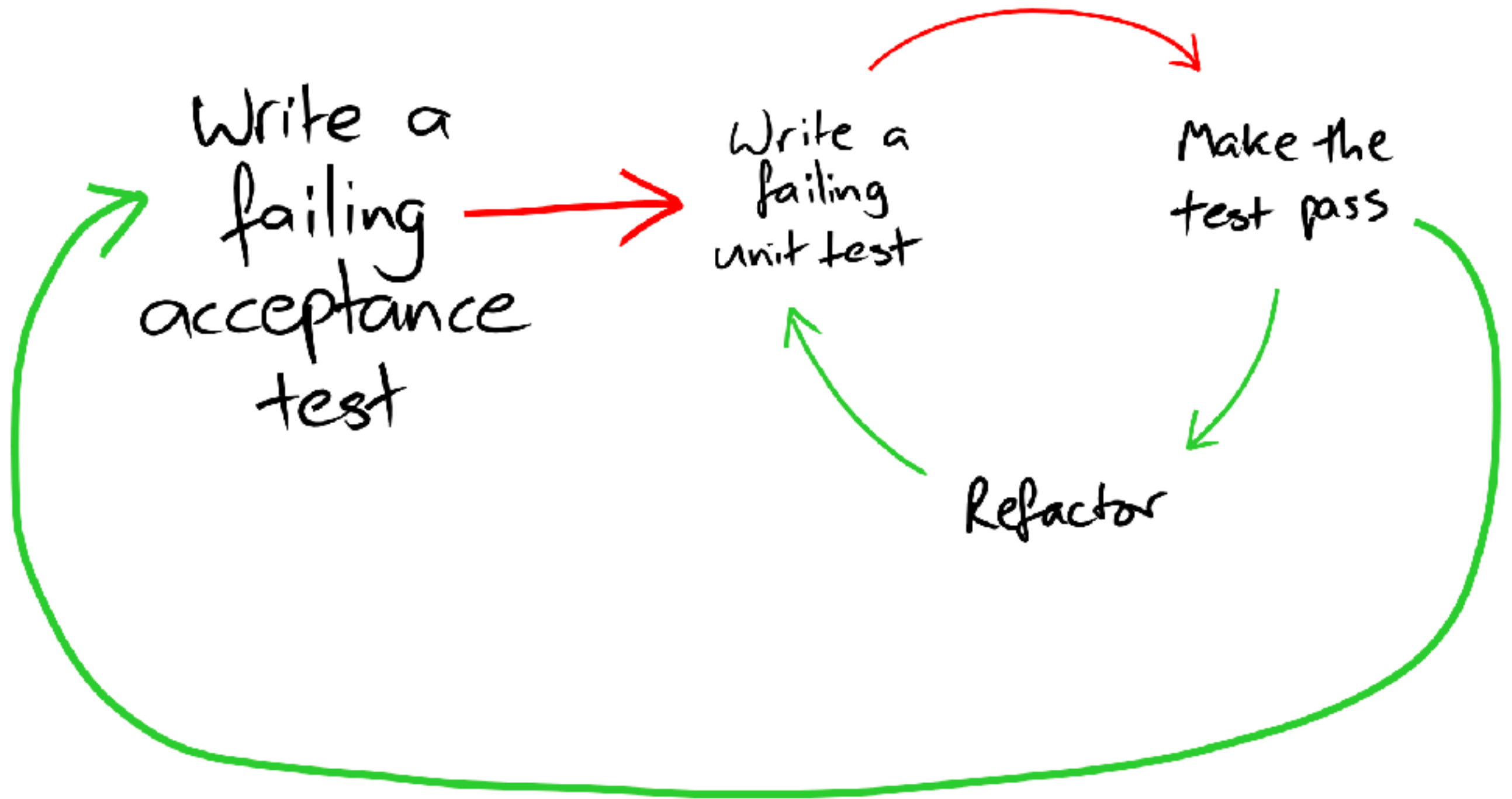
Start with **simple**



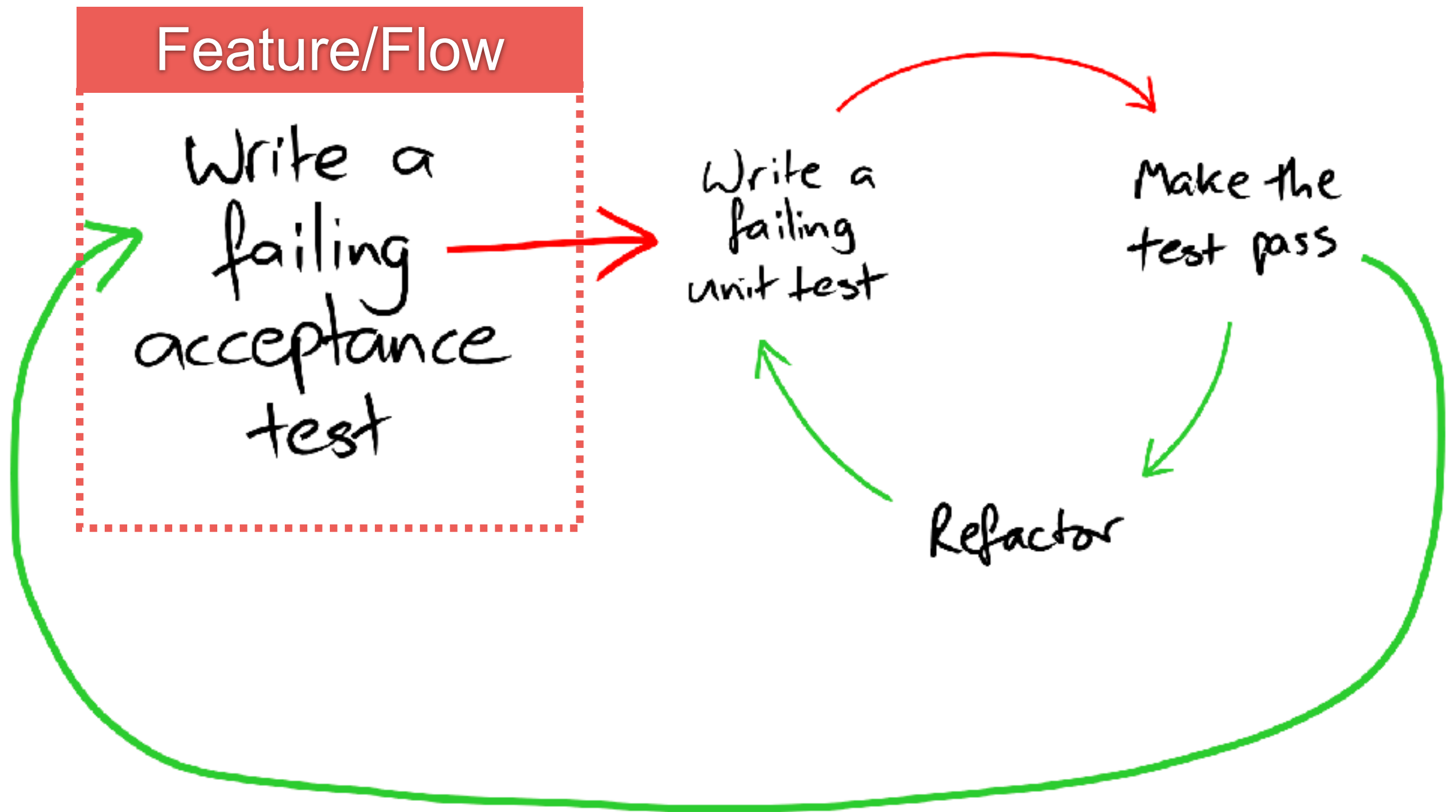
Use **feedback** to improve



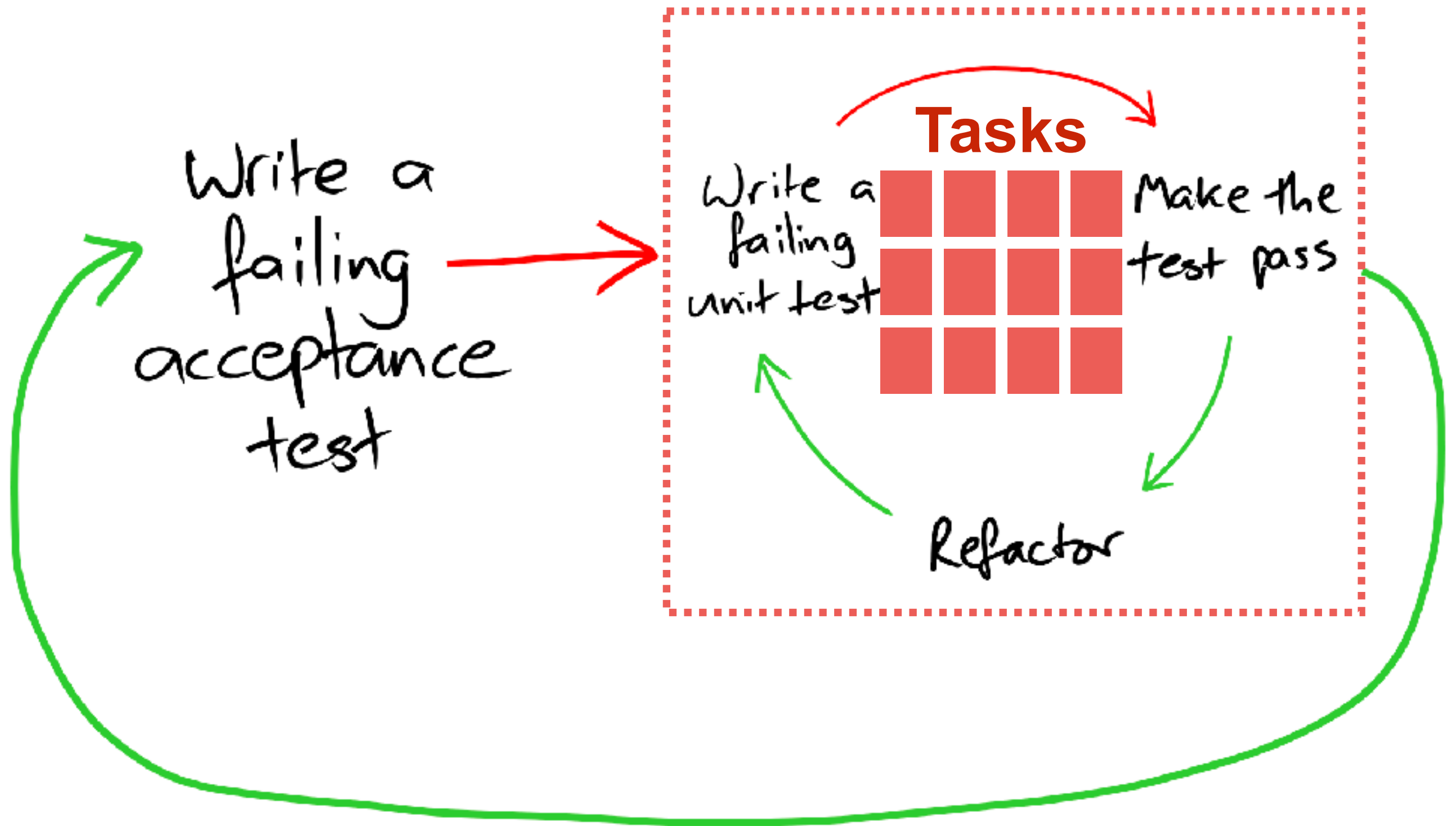
Outside-in develop/test



Outside-in develop/test



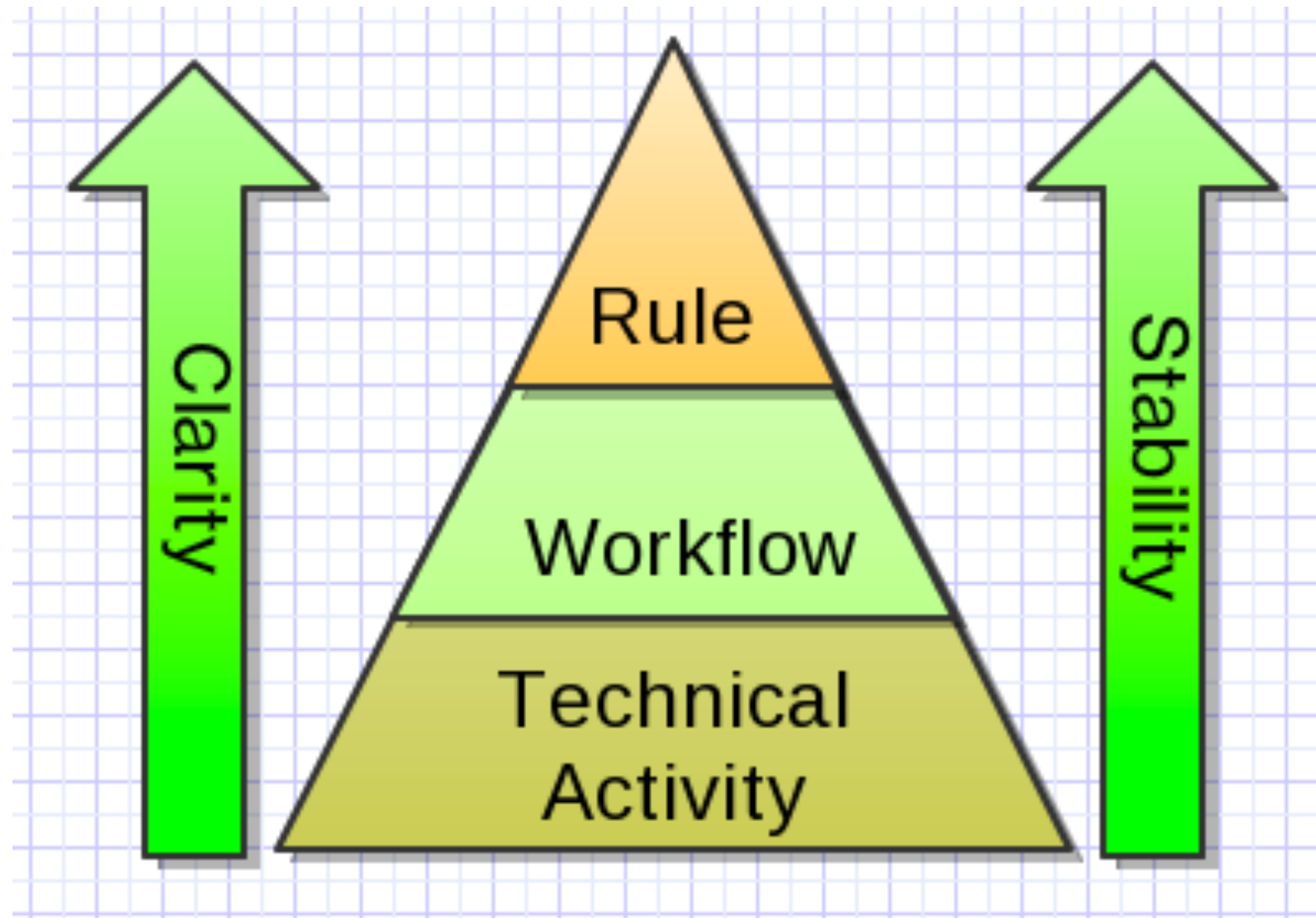
Outside-in develop/test



UI Test Automation



3 levels of UI test automation



3 levels of UI test automation

Business rule/functionality level

what is this test demonstrating or exercising

User interface workflow level

what does a user have to do to exercise the functionality through the UI

Technical activity level

what are the technical steps required to exercise the functionality



Finding and Using Elements



Appium Tools

Appium Server via npm
Appium Server GUI/Desktop
Appium Inspector

<http://appium.io/docs/en/about-appium/getting-started/?lang=en#installing-appium>



Locator Strategies

Strategy	Description
Accessibility ID	iOS = accessibility-id Android = content-desc
ID	iOS = name Android = resource-id
Name	Name of element
XPath	Pattern of element in XML (not recommended)
Class name	

<http://appium.io/docs/en/commands/element/find-elements/index.html#selector-strategies>



Good Locators

Unique
Descriptive
Resilient

Shorter in length (maintainability)



XPath

CML Path Language

Query language used to identify tag in XML
Appium builds XML representation of app

XPath is powerful by very dangerous



Benefits of XPath

Find any element that exists

Find elements by using complex criteria



Cons of XPath

XPath selectors can be brittle

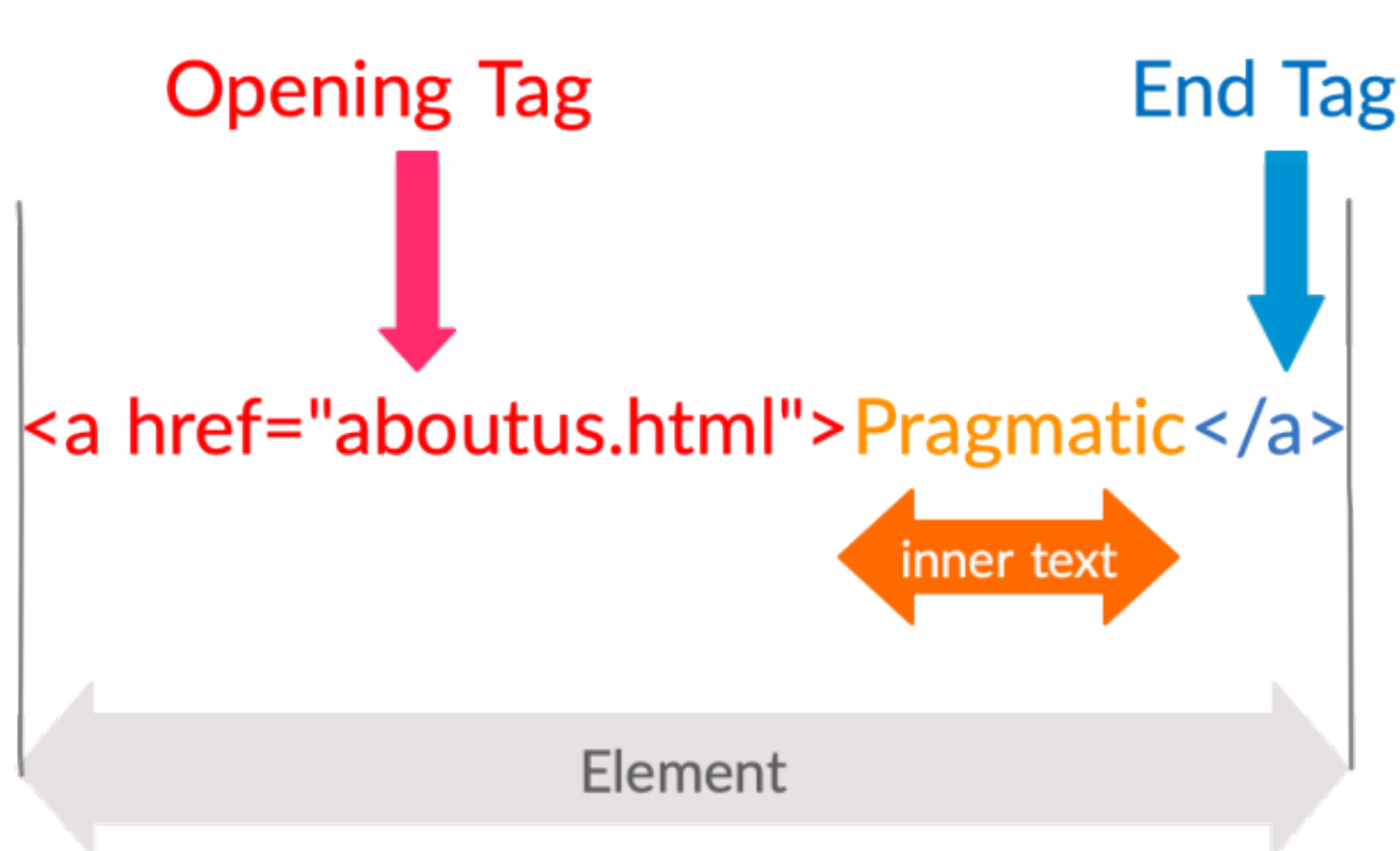
Slow

Broken test !!

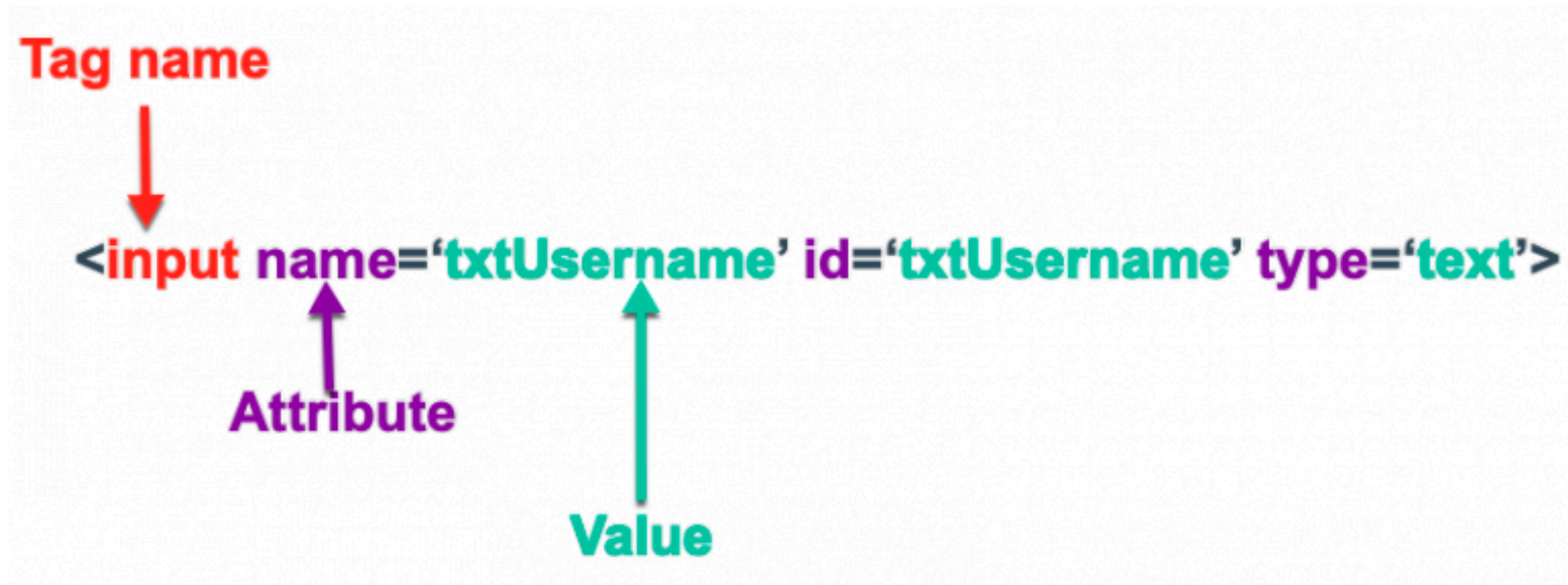
Avoid brittle selectors by rely on unique tag attribute



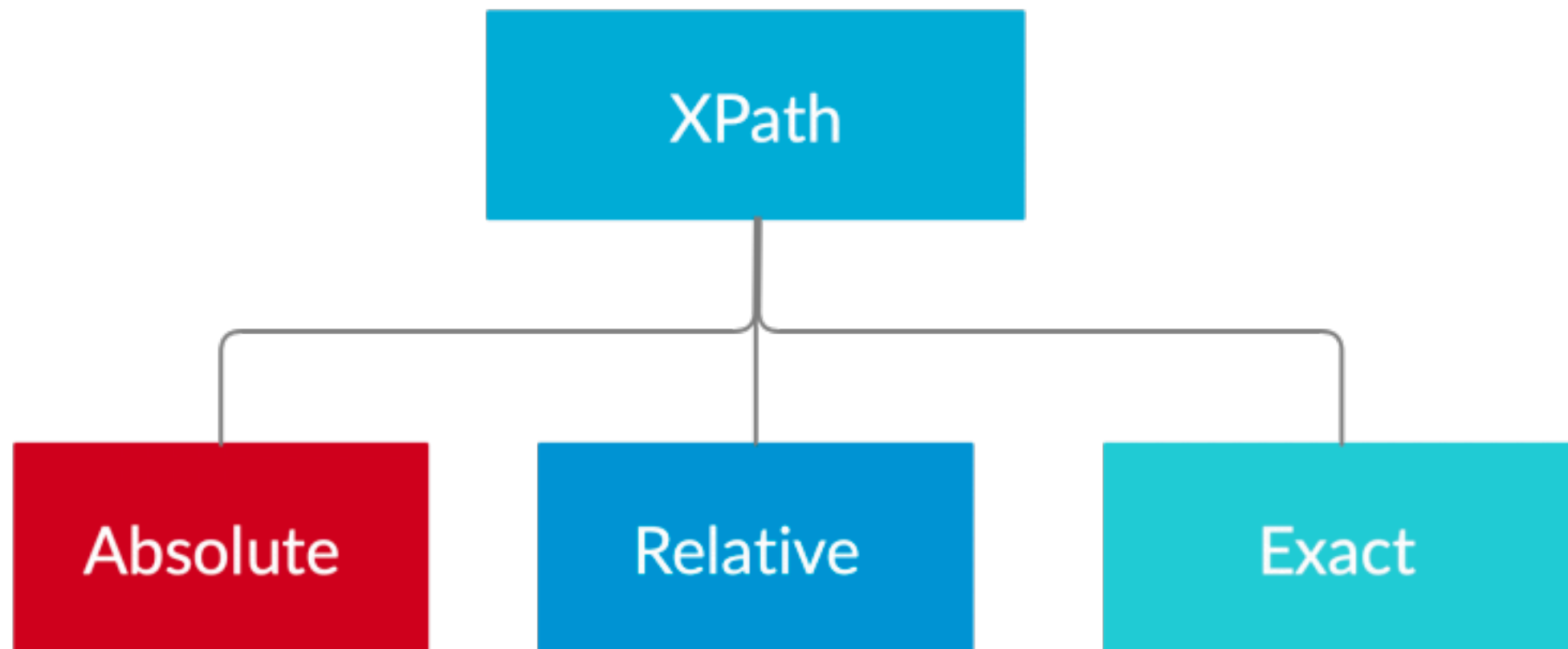
XPath



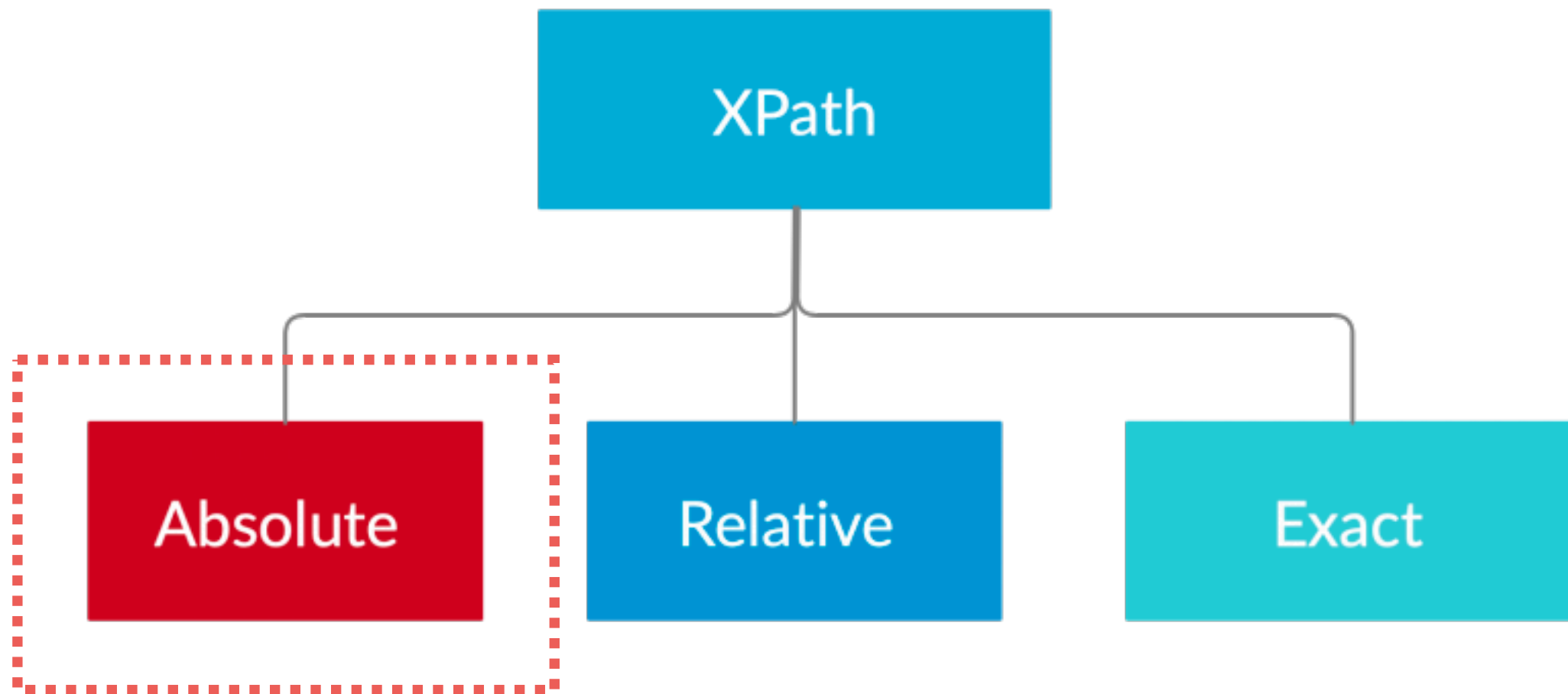
XPath



Types XPath



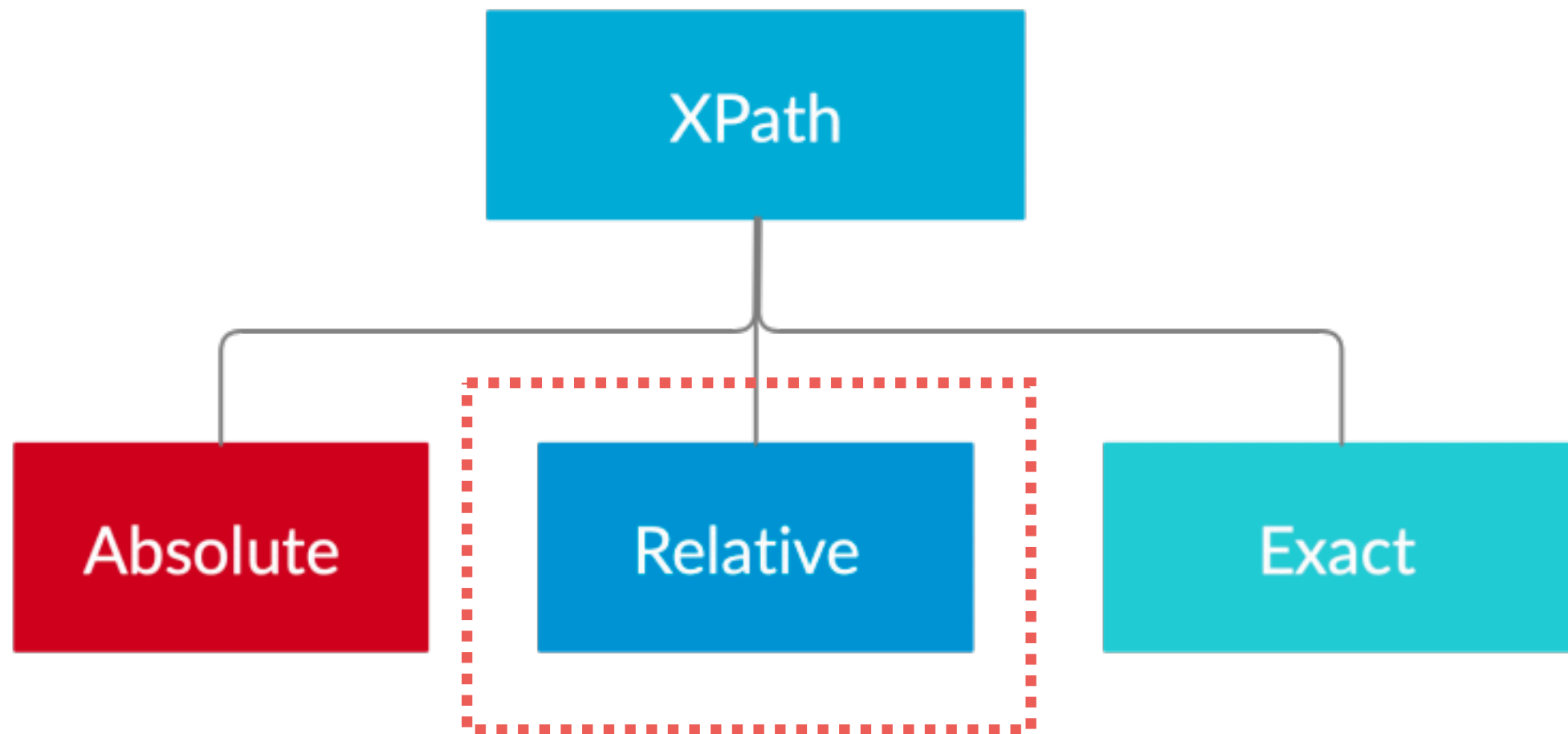
Don't use !!



`/html/body/div[1]/div/div[2]/form/div[2]/input`



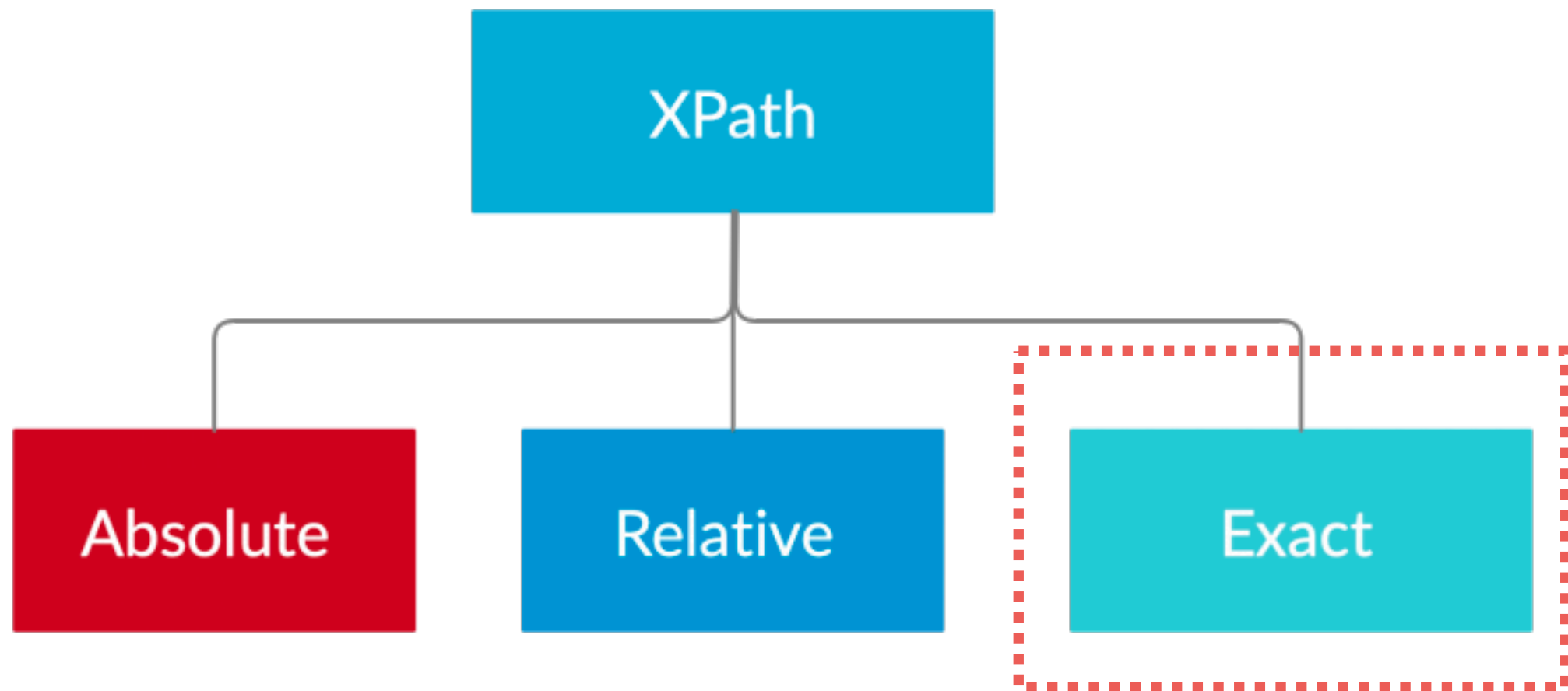
Relative



//div[@id='divUsername']/input
//form/div[@id='divUsername']/input
//form/*/input



Exact



`//div[@class='datevalue currmonth']//span[./text()='2']`



Absolute faster than Relative
But shortest is better



Element Interactions



Element Interactions

Command	Description
click()	Tab an element
sendKeys()	Enter keystrokes into an input field
clear()	Clear an input field
getText()	Retrieve the text displayed from a field or label

<http://appium.io/docs/en/commands/element/find-elements/index.html#selector-strategies>



Waiting for Elements



Waiting ...

Static waits (sleep)

Explicit waits



Working with Robot Framework



<https://robotframework.org/>





<https://github.com/serhatbolsu/robotframework-appiumlibrary>



Install via pip

```
$pip install --upgrade robotframework
```

```
$pip install --upgrade robotframework-appiumlibrary
```



Manage App's States

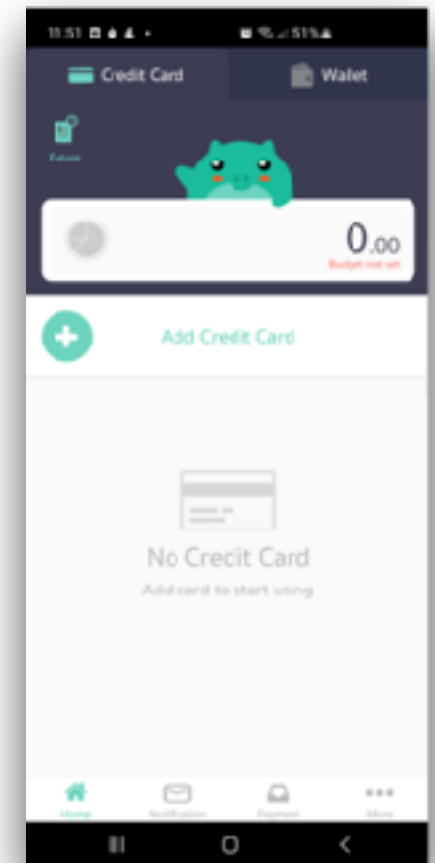
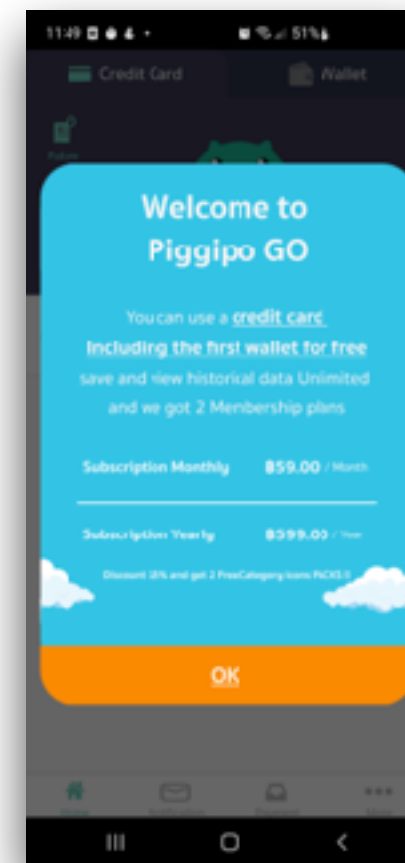
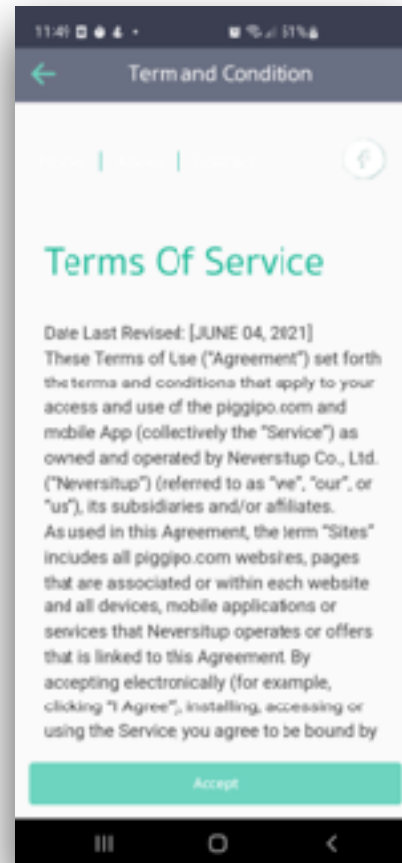
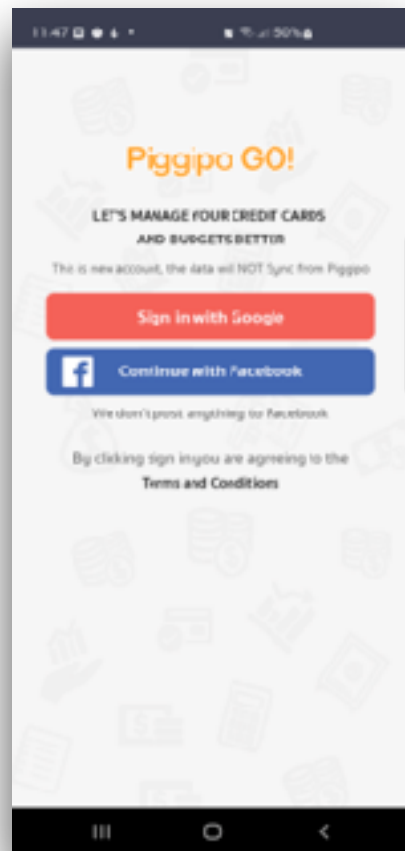
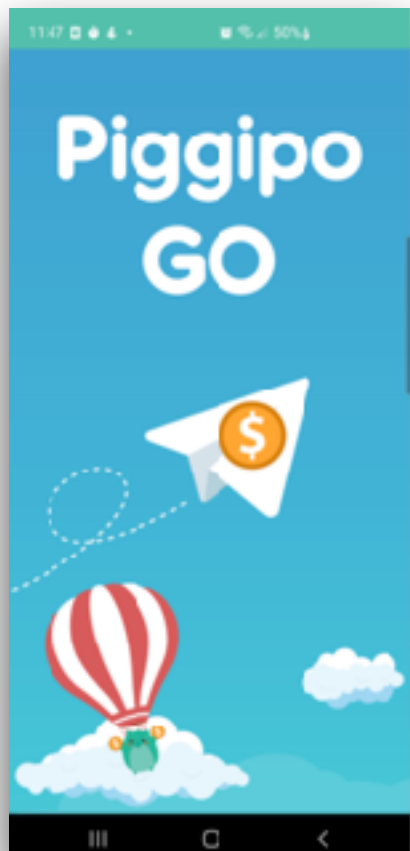


Manage App's States

Easy for test
Reduce test steps



Manage App's States



Manage App's States

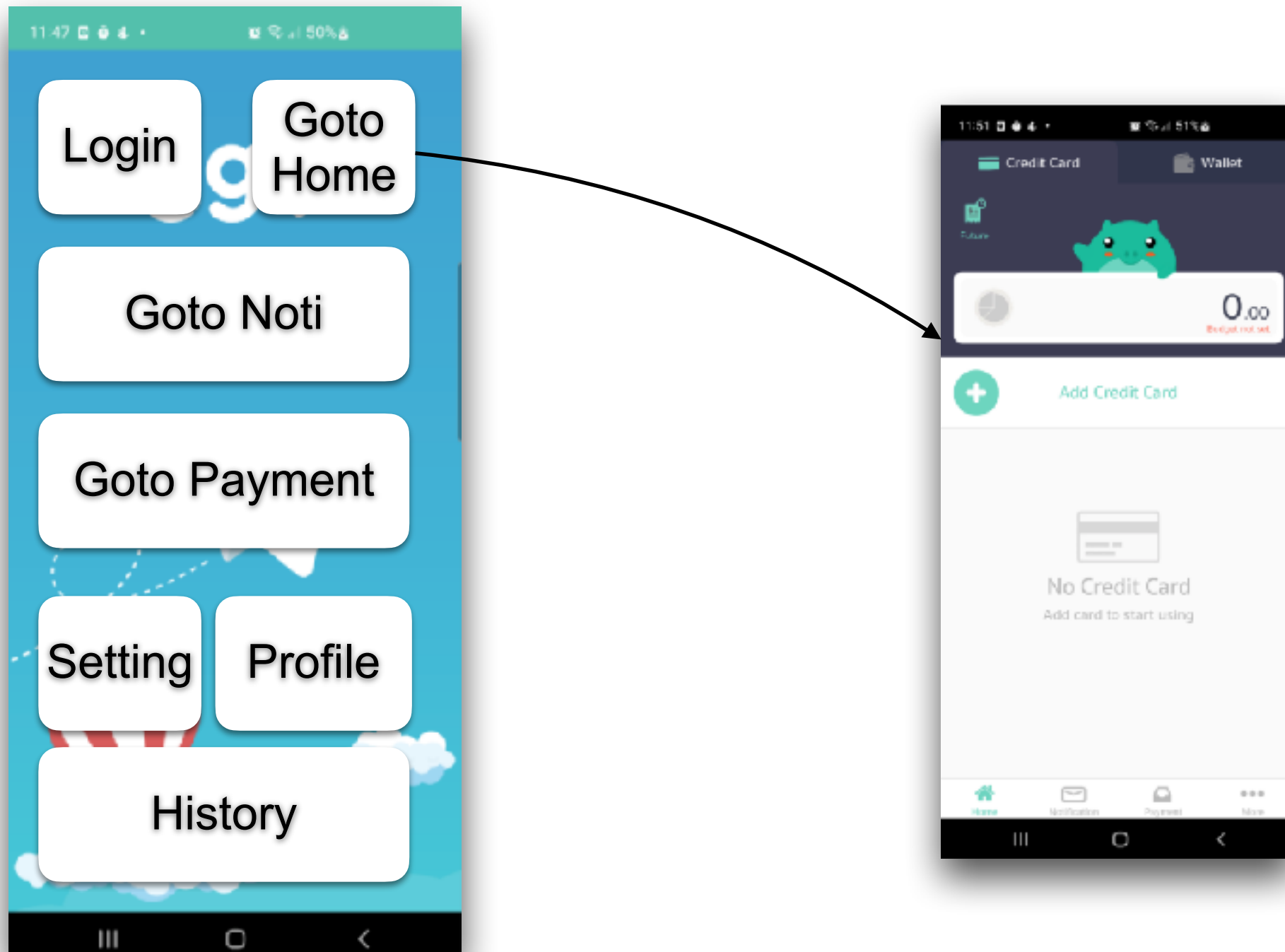
Use only in test build of the app

Screen with on-click link to all area of app

Link can even populate data



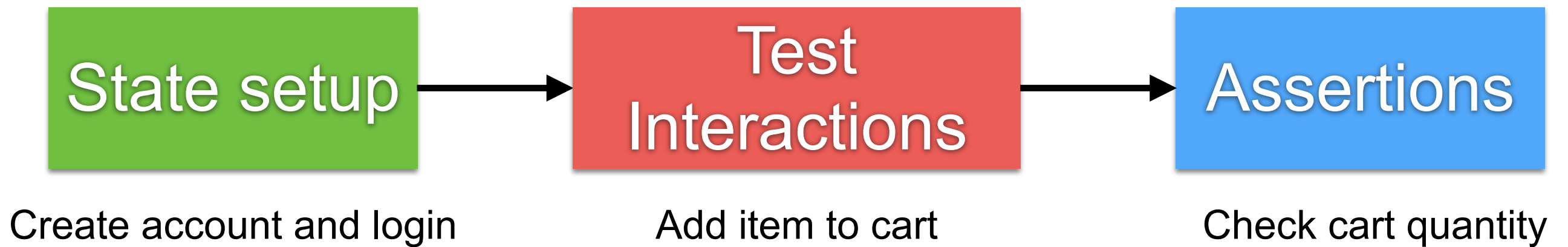
Manage App's States



State vs. Assertions

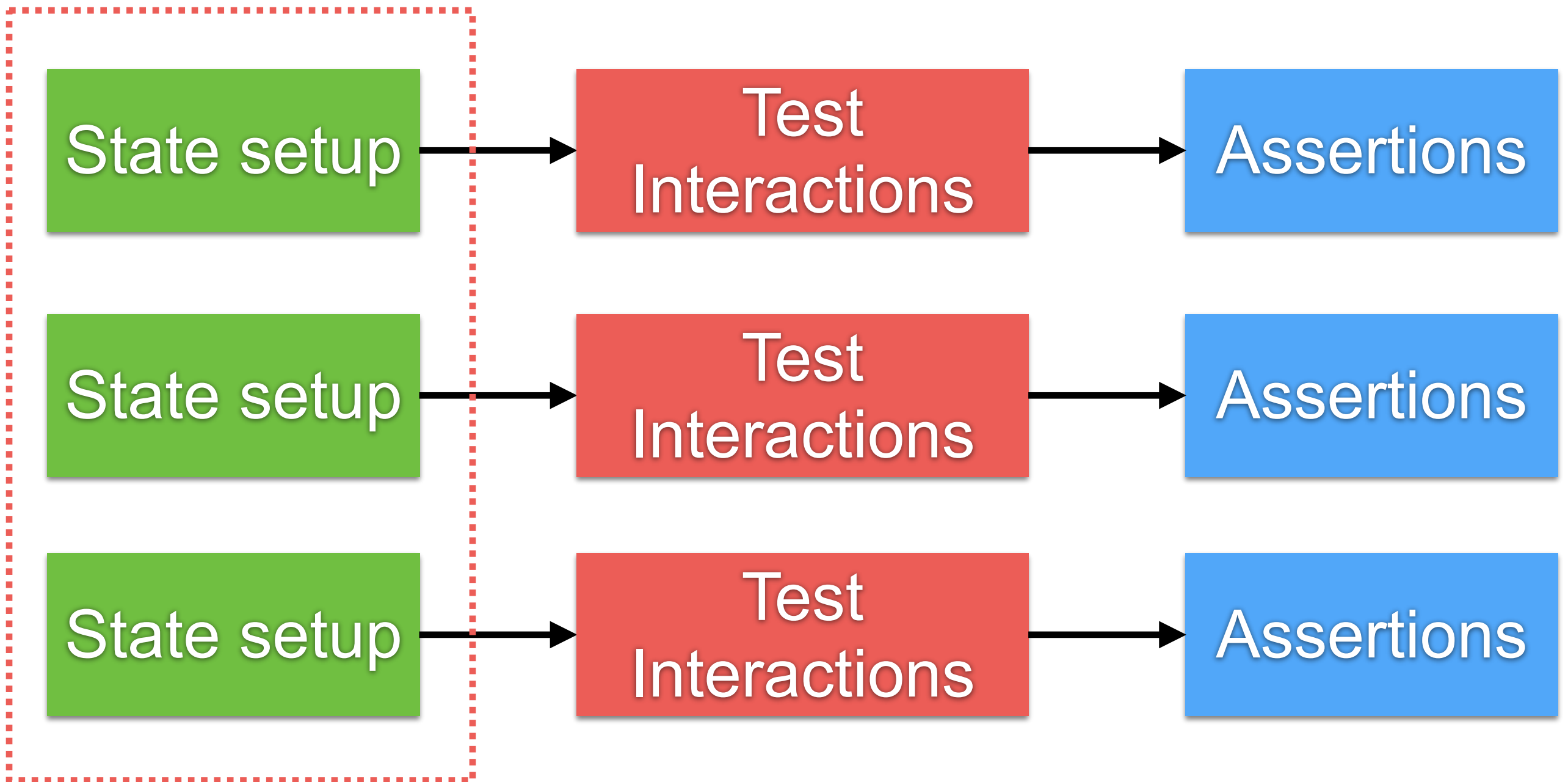


State vs. Assertions



State vs. Assertions

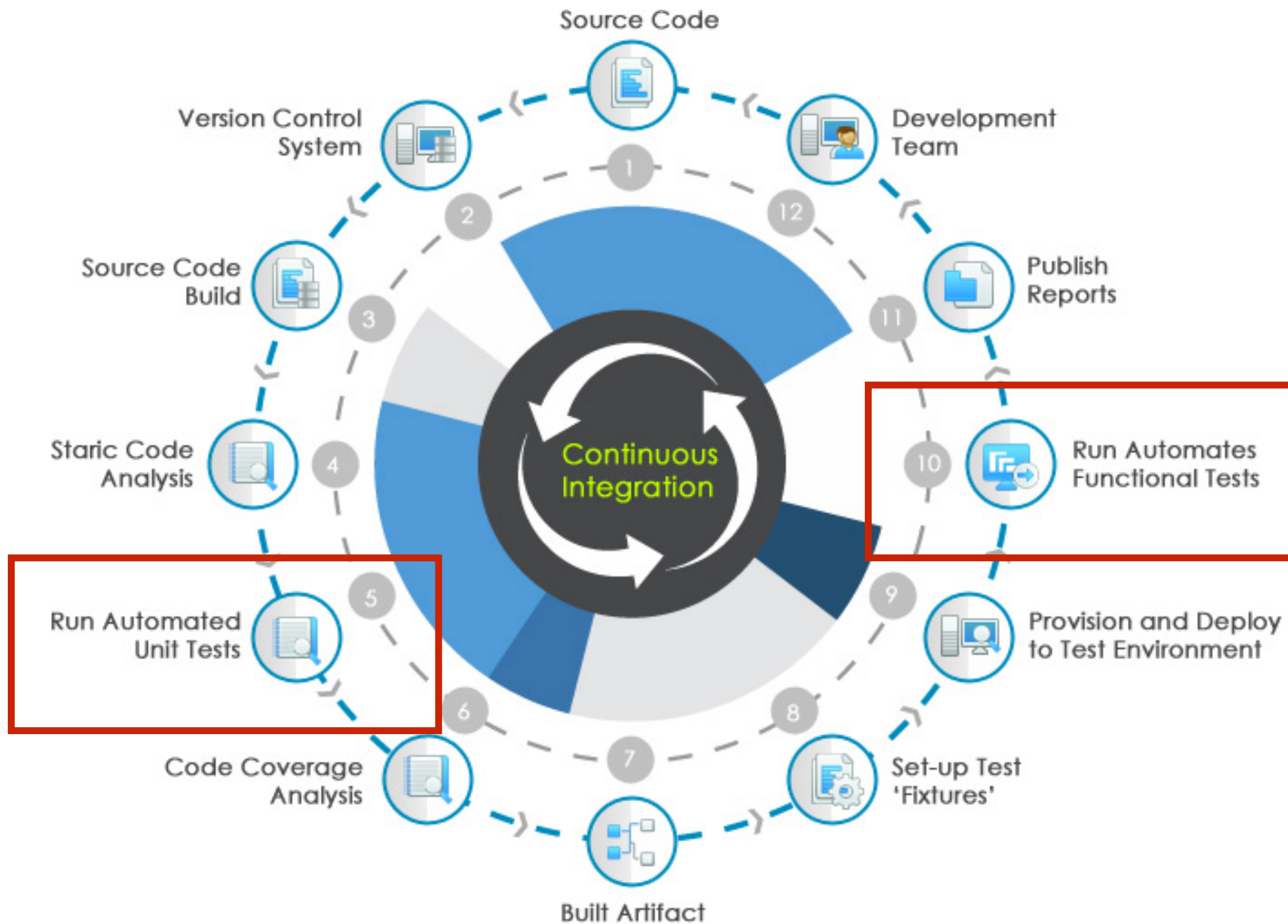
Not too much time !!



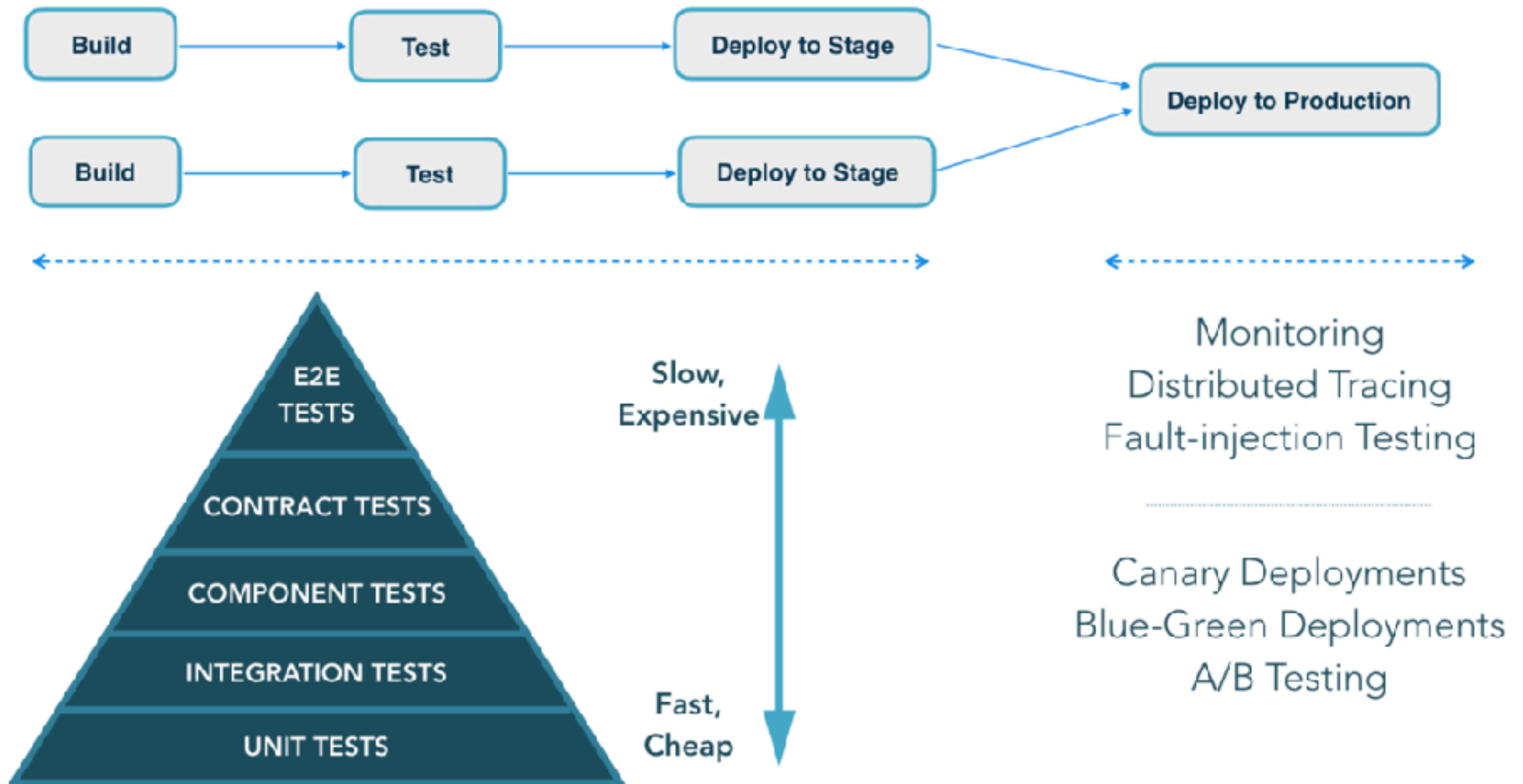
Continuous Integration



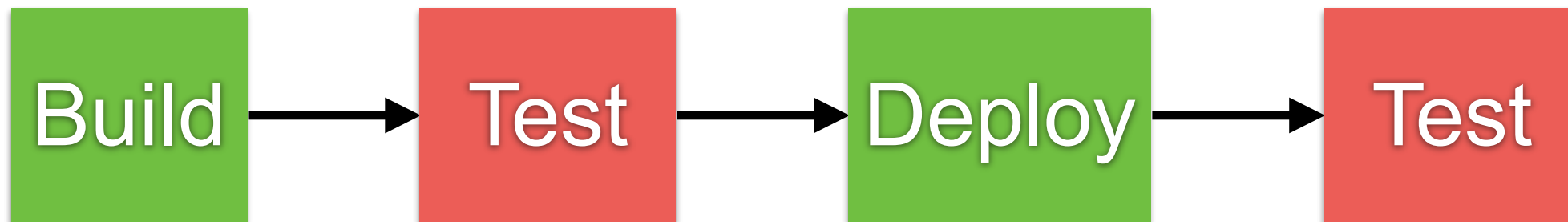
Continuous integration



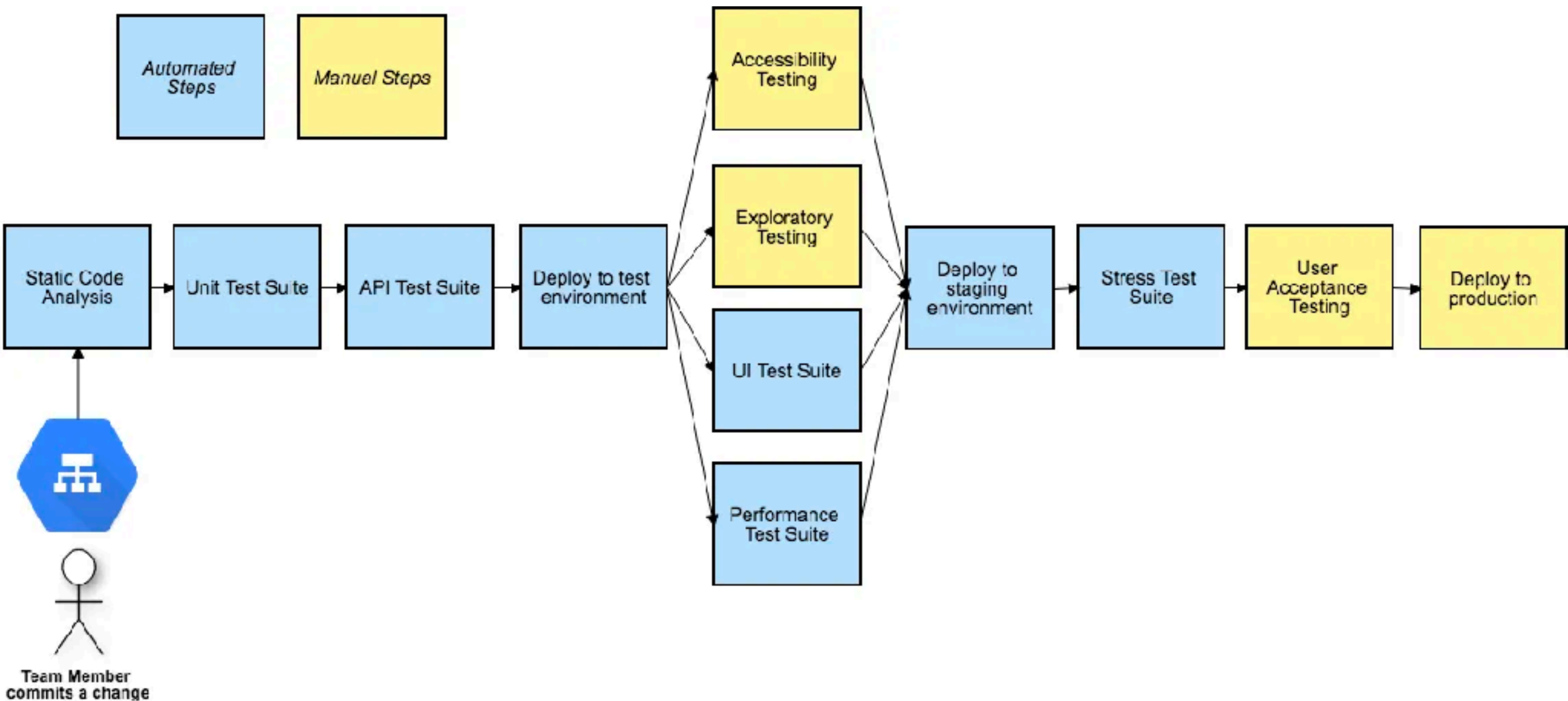
Test strategy



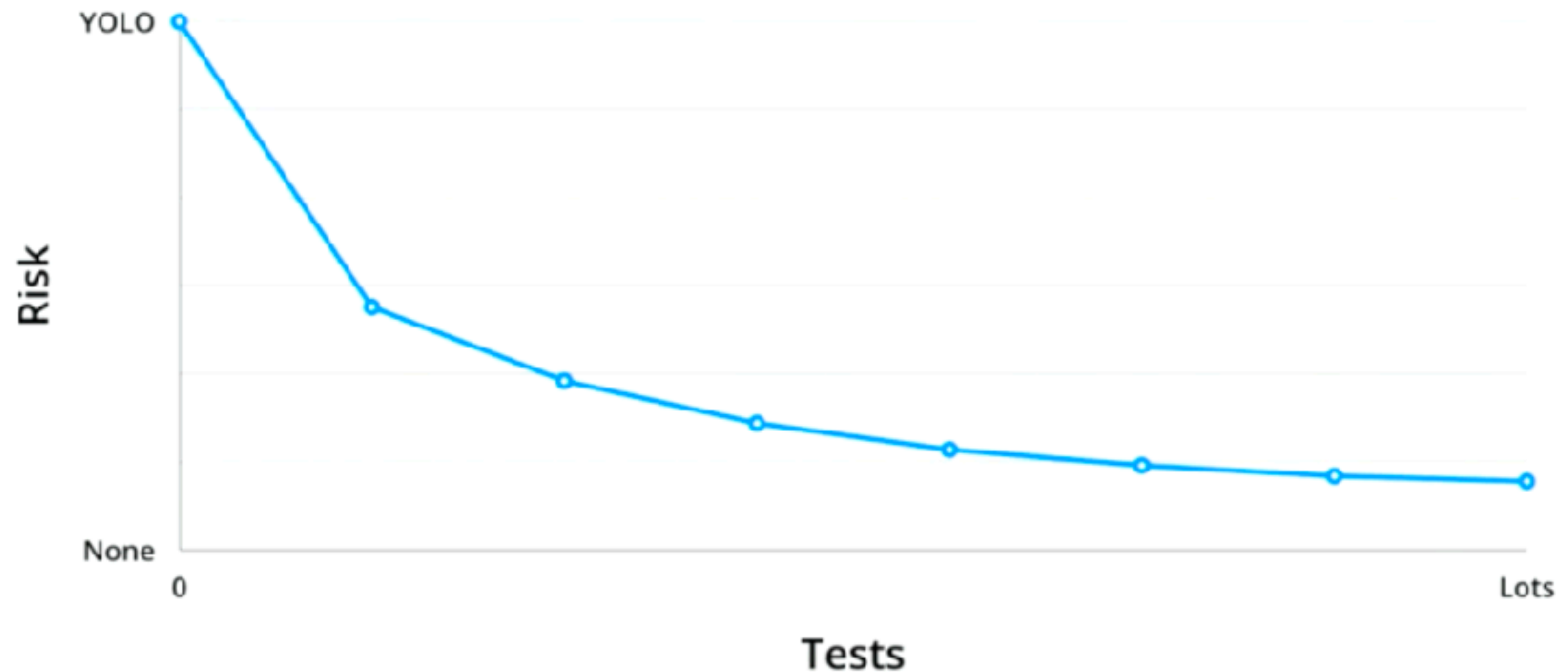
Design your pipeline



Design your pipeline/process



Reduce risk with tests



Q/A

