



Mobile Automated Testing with Appium



Automated Testing



<https://appium.io/docs/en/latest/>



**[https://github.com/up1/
course-appium-robotframework](https://github.com/up1/course-appium-robotframework)**



Topics

Think about Test !!
Mobile testing strategies
Introduction to Appium
Working with **Android**, iOS and Flutter app
Write test cases
Tips and techniques



Test cases with Appium

Java

.NET C#

JavaScript

Python

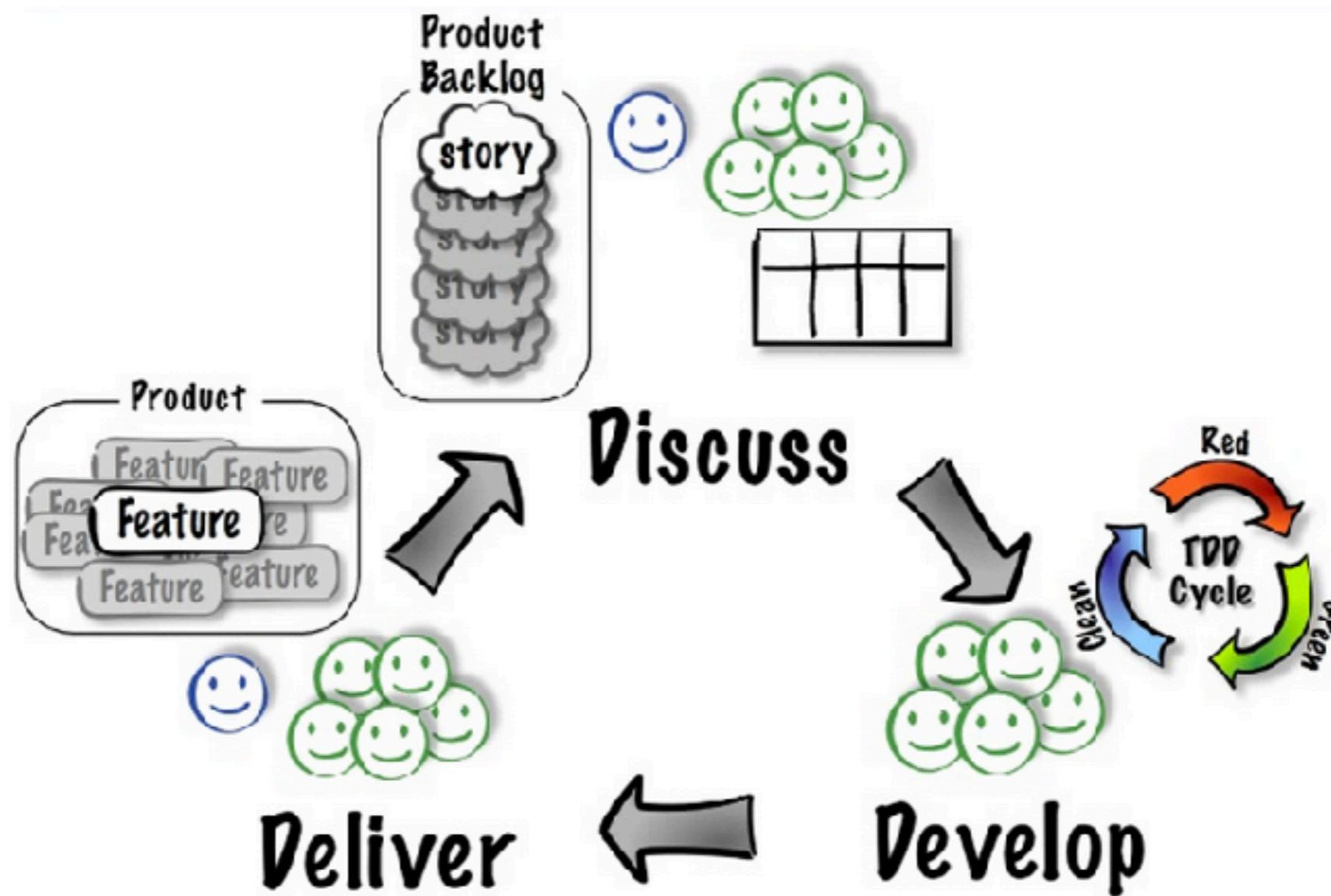
Robotframework + AppiumLibrary



THINK before coding



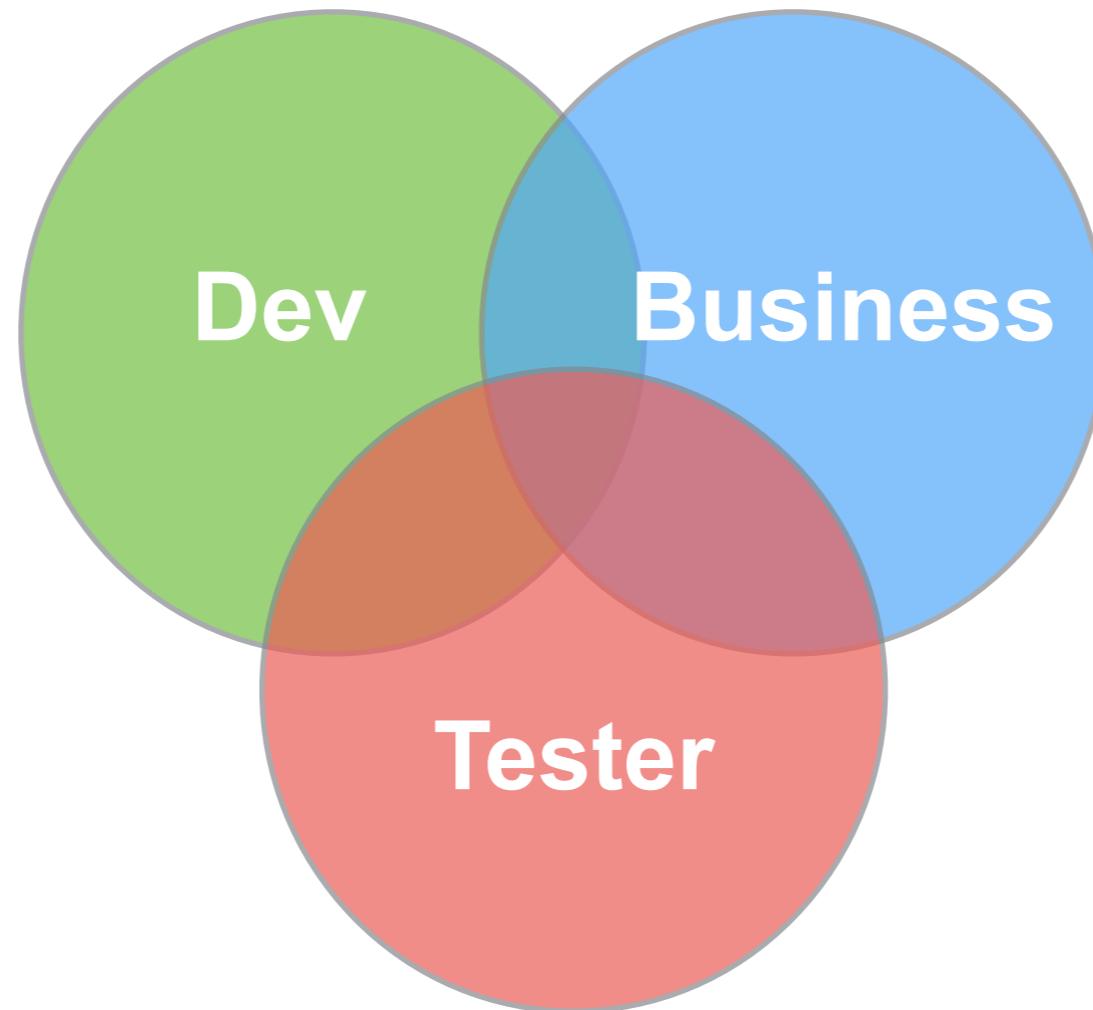
Acceptance Test-Driven Development



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



Acceptance Test-Driven Development



Acceptance Tests

=

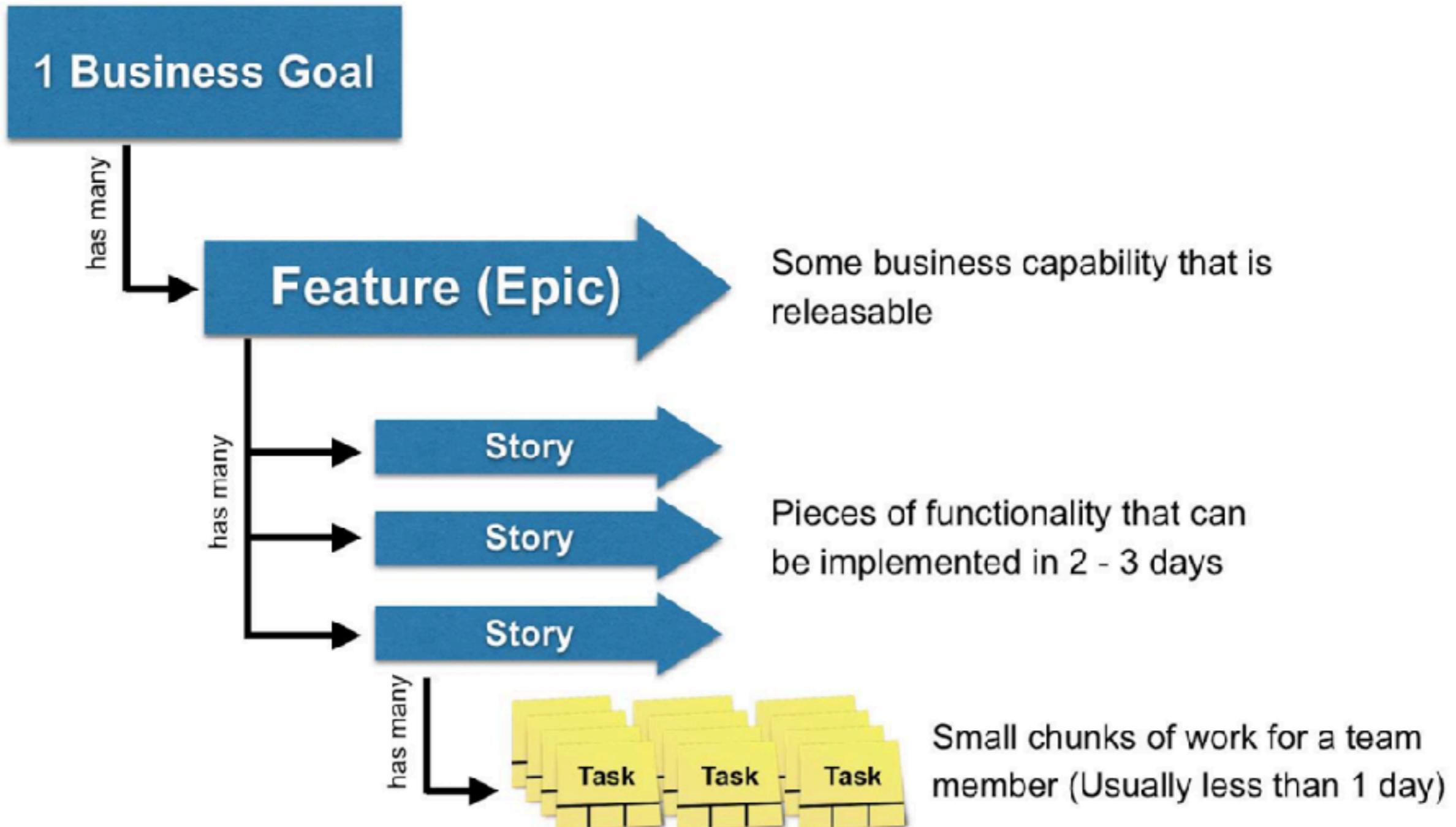
Business Criteria

+

Examples (data)



Work break down



Iterative and incremental process

Feature 1

Time



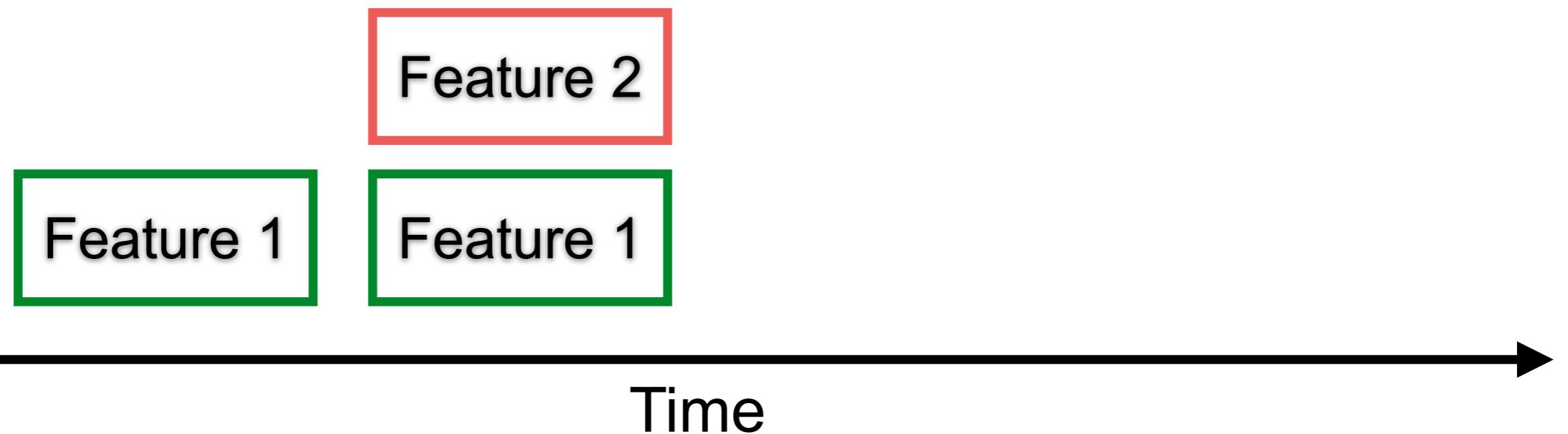
Iterative and incremental process

Done = coded and tested



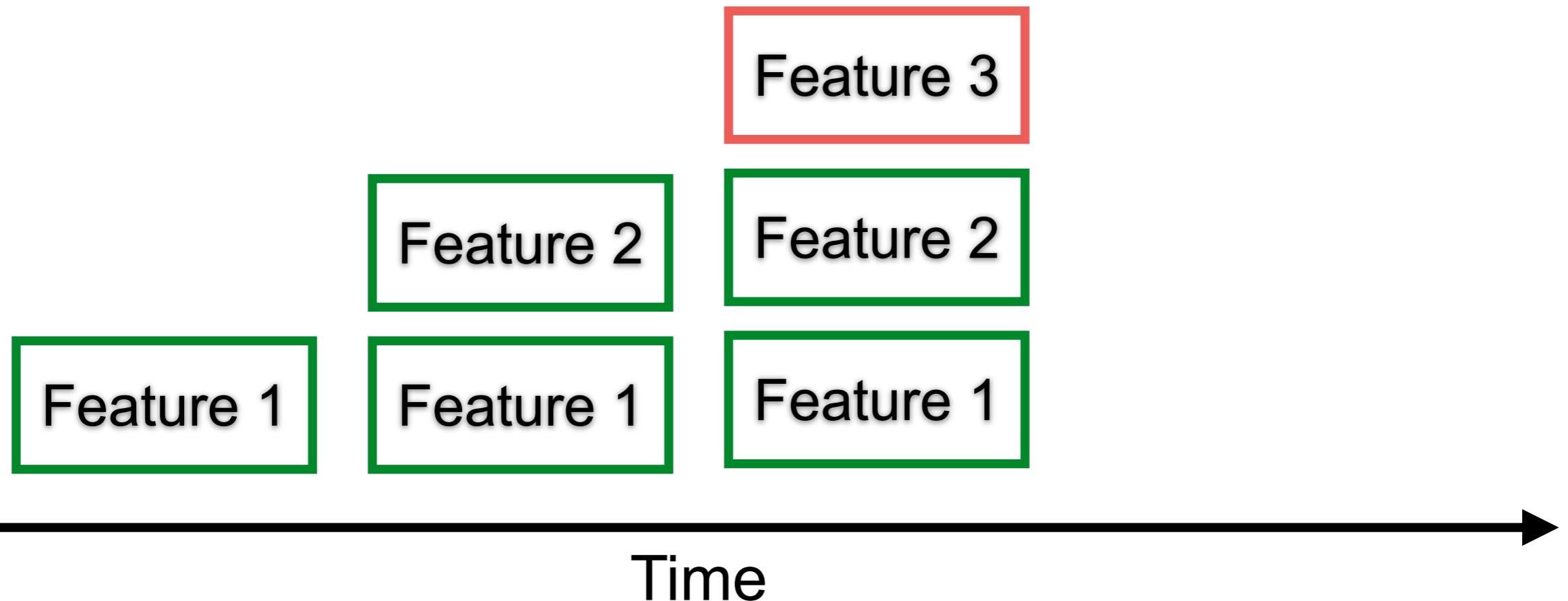
Iterative and incremental process

Done = coded and tested



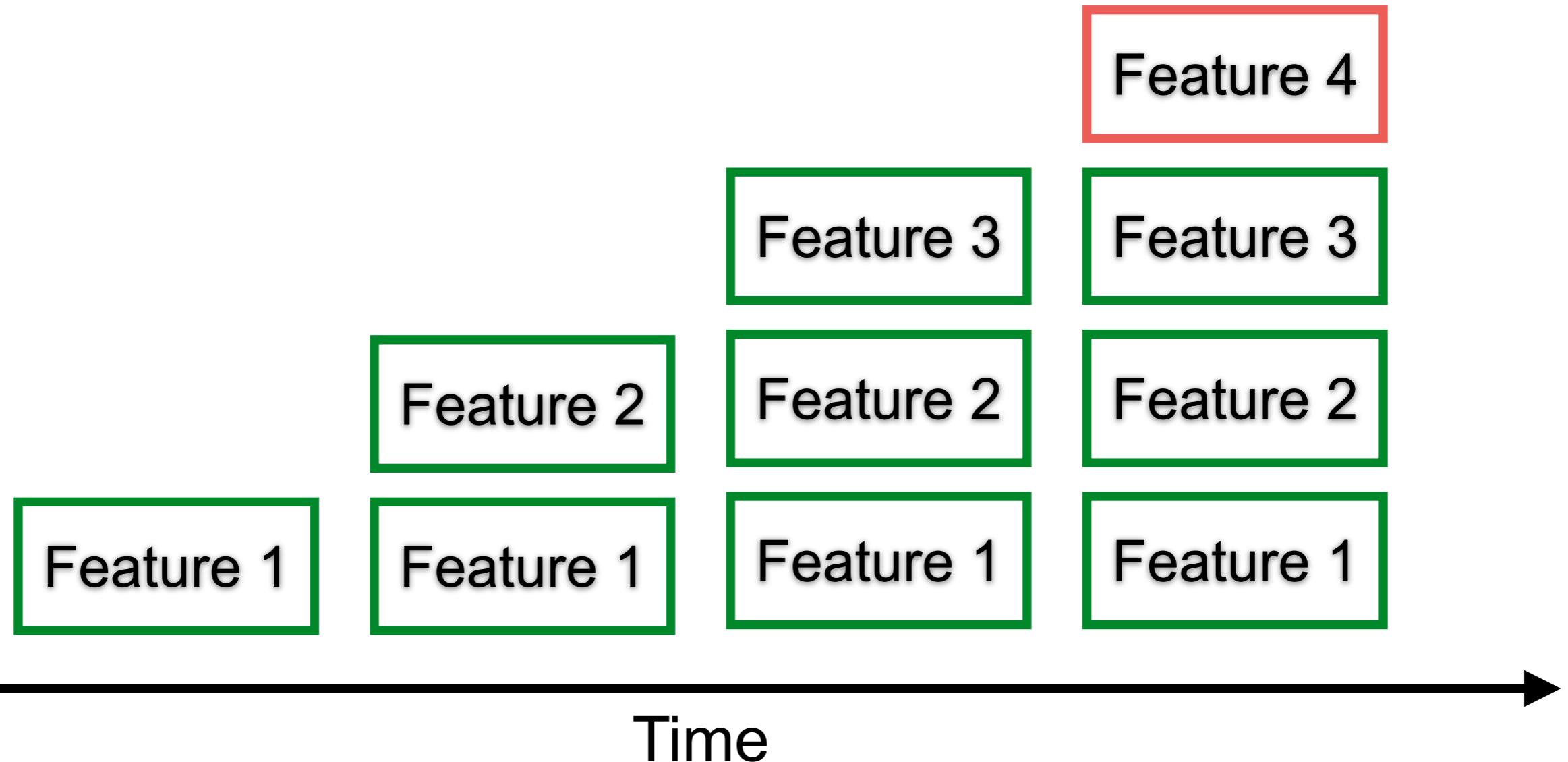
Iterative and incremental process

Done = coded and tested



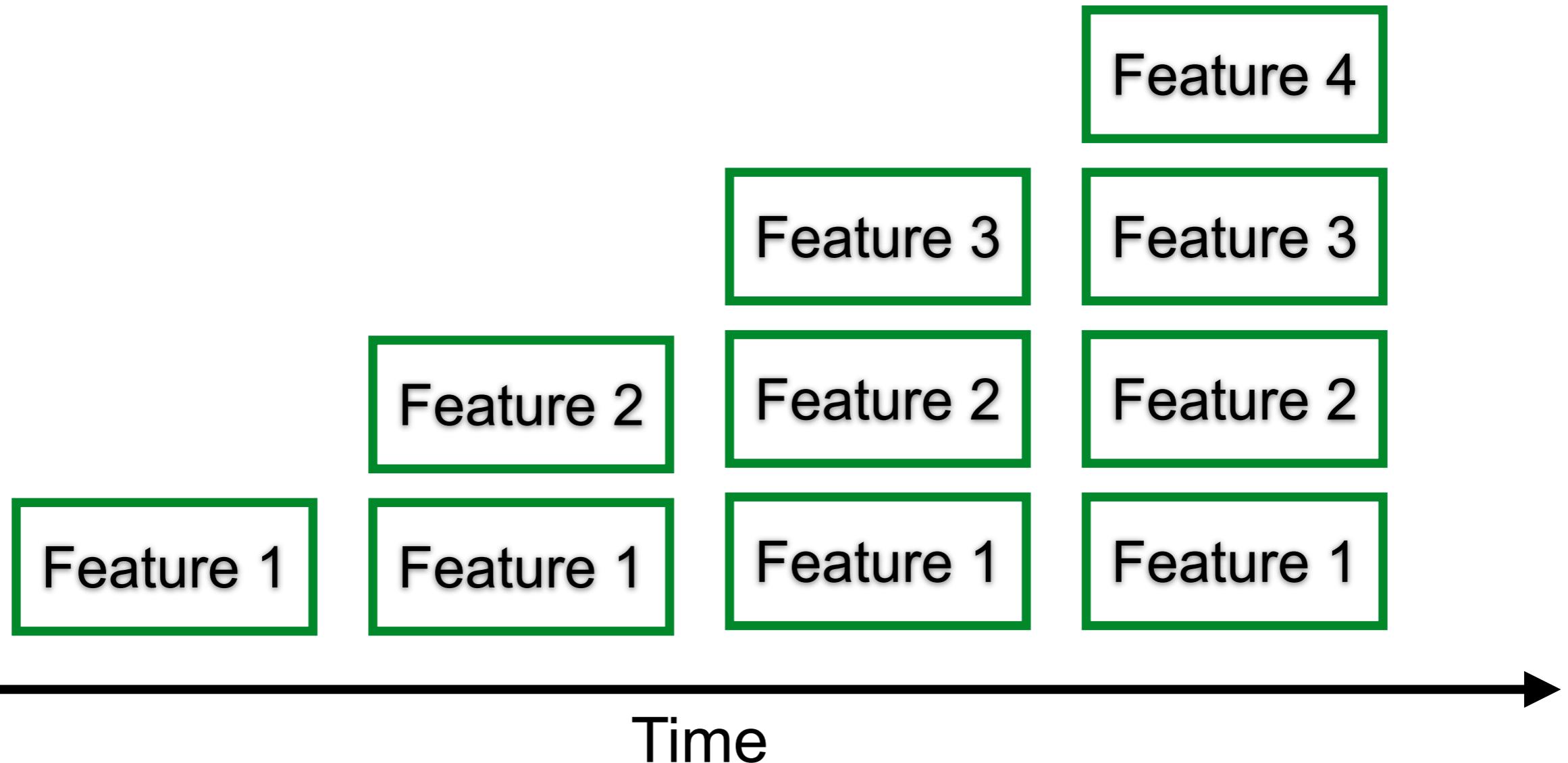
Iterative and incremental process

Done = coded and tested



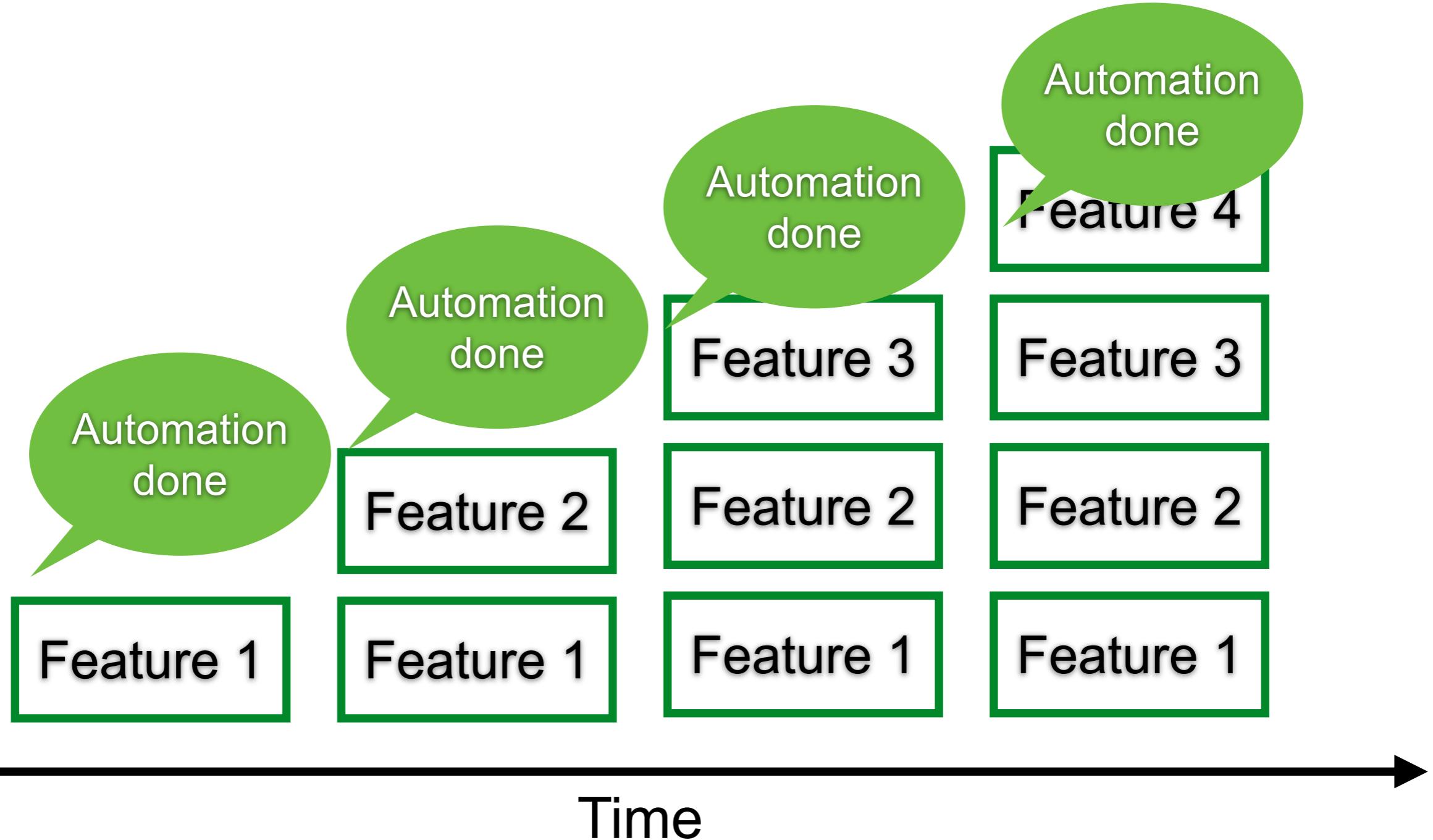
Iterative and incremental process

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 ?



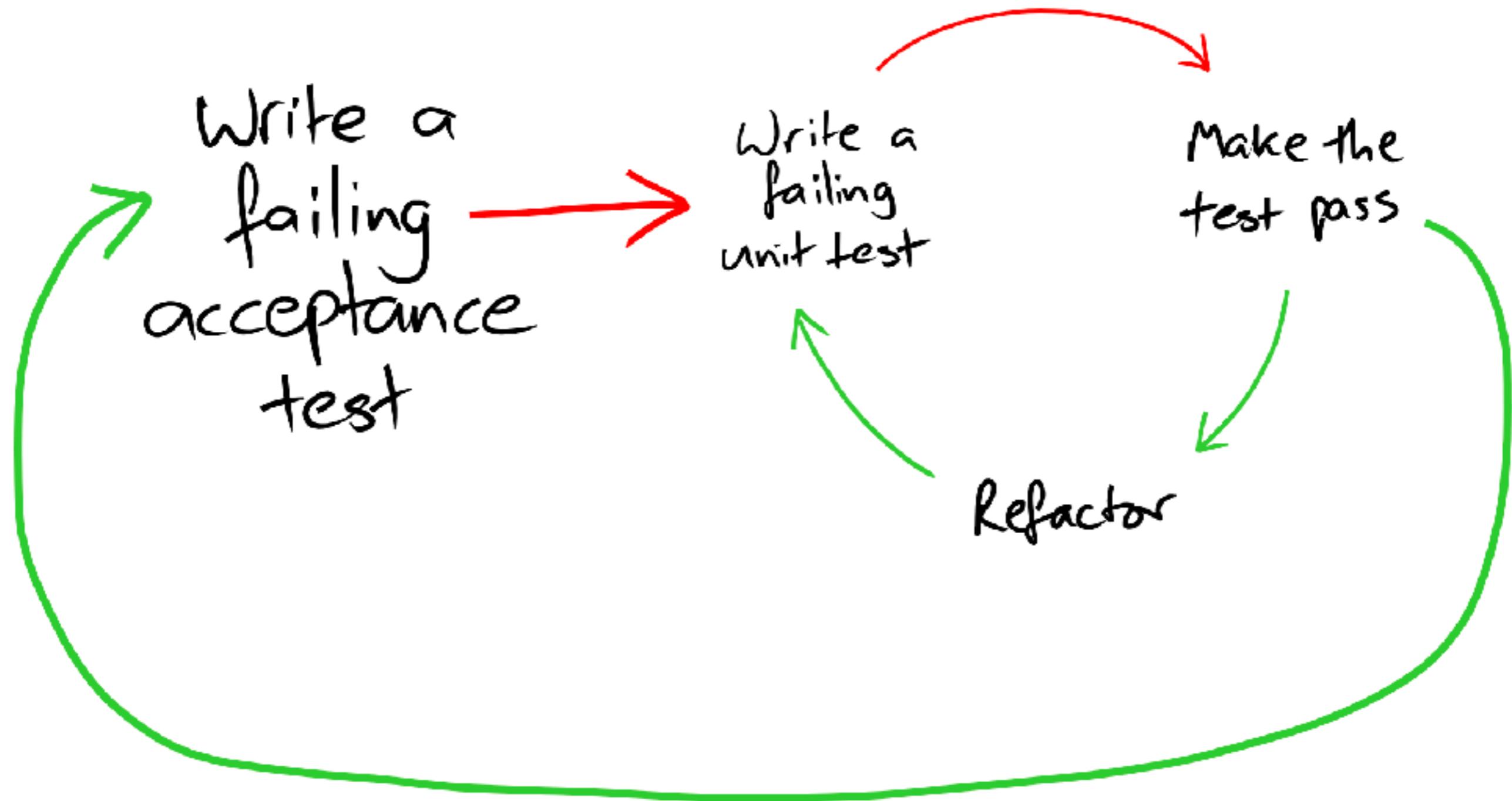
Start with simple



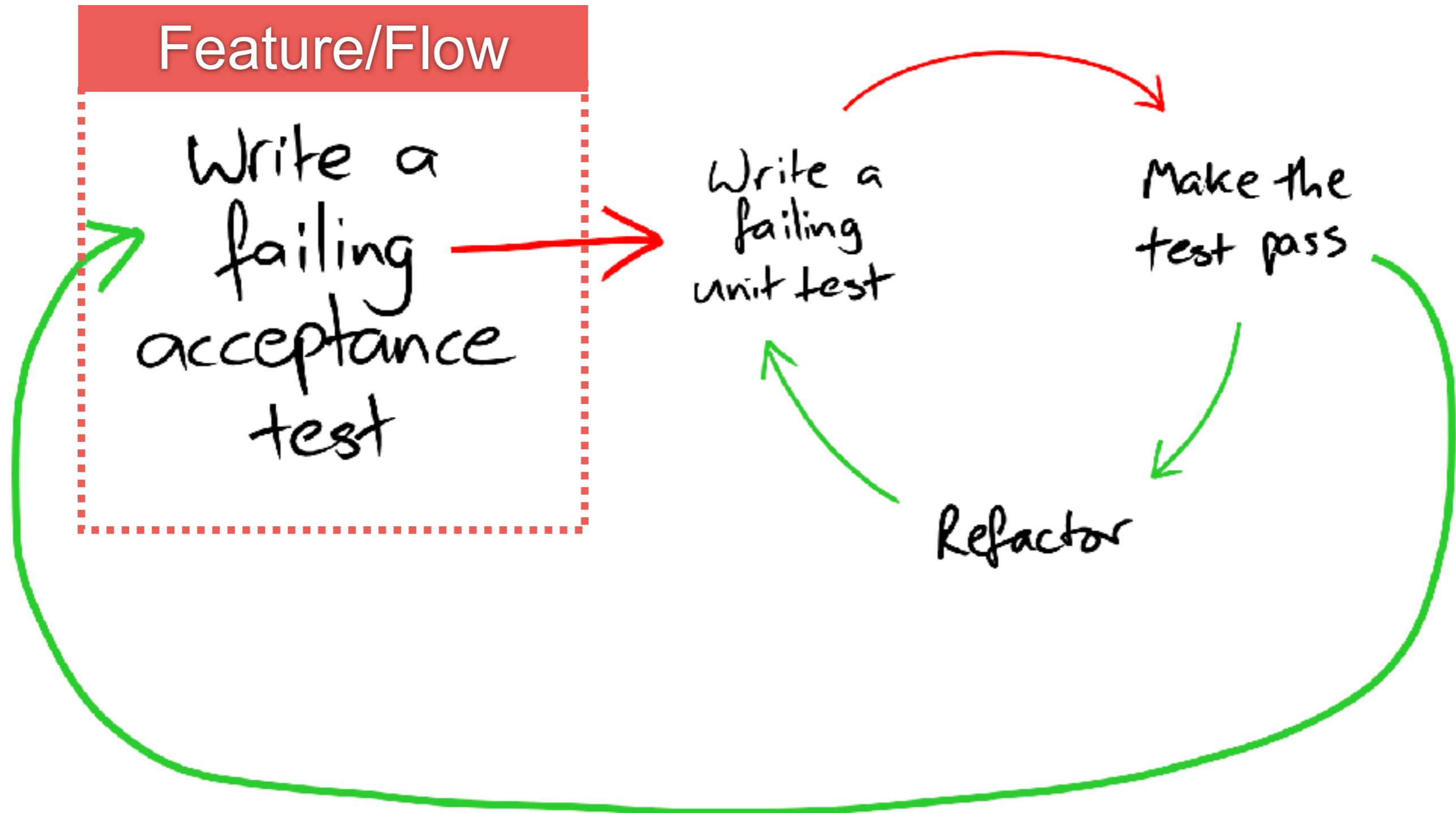
Use **feedback** to improve



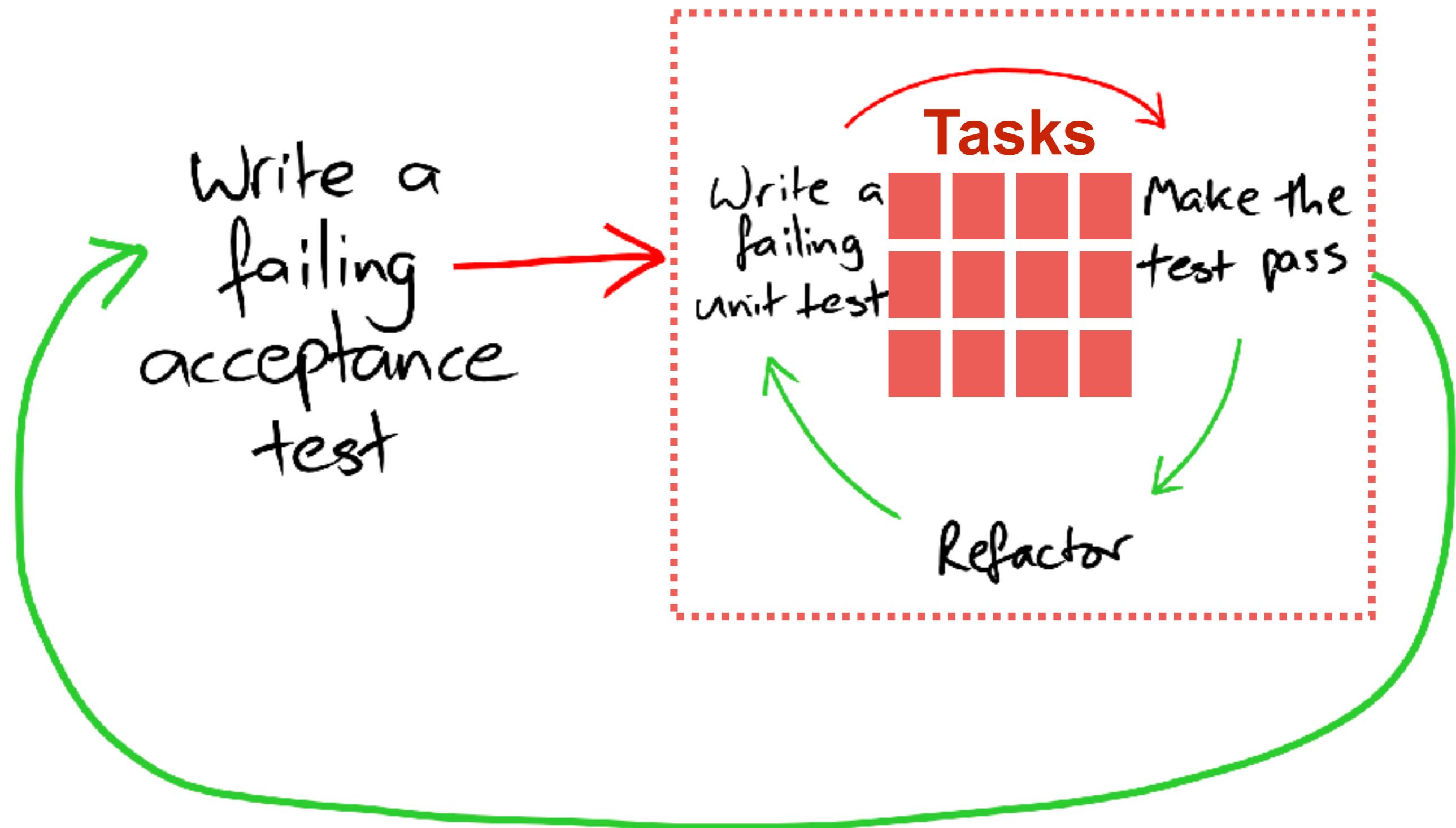
Outside-in develop/test



Outside-in develop/test



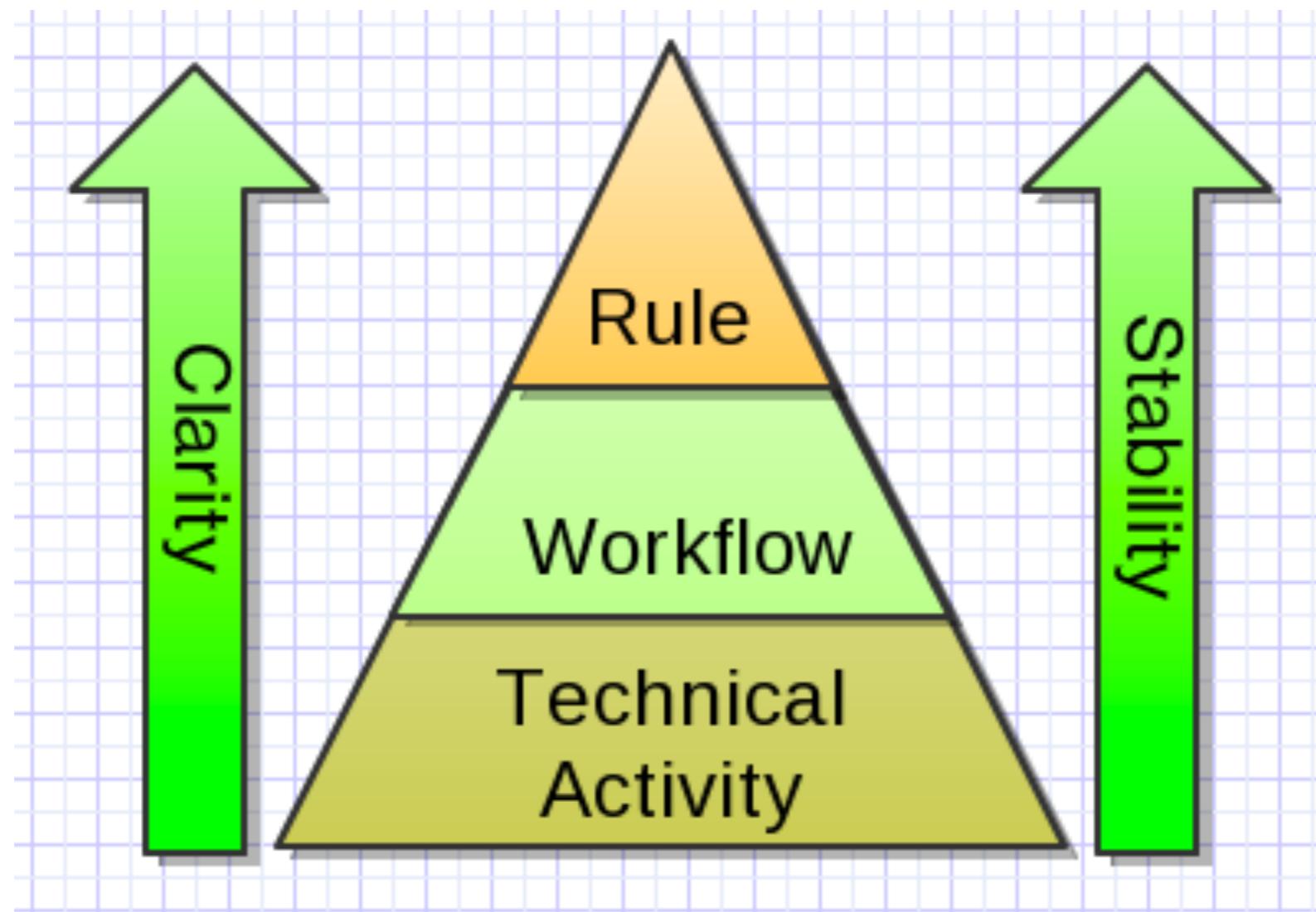
Outside-in develop/test



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

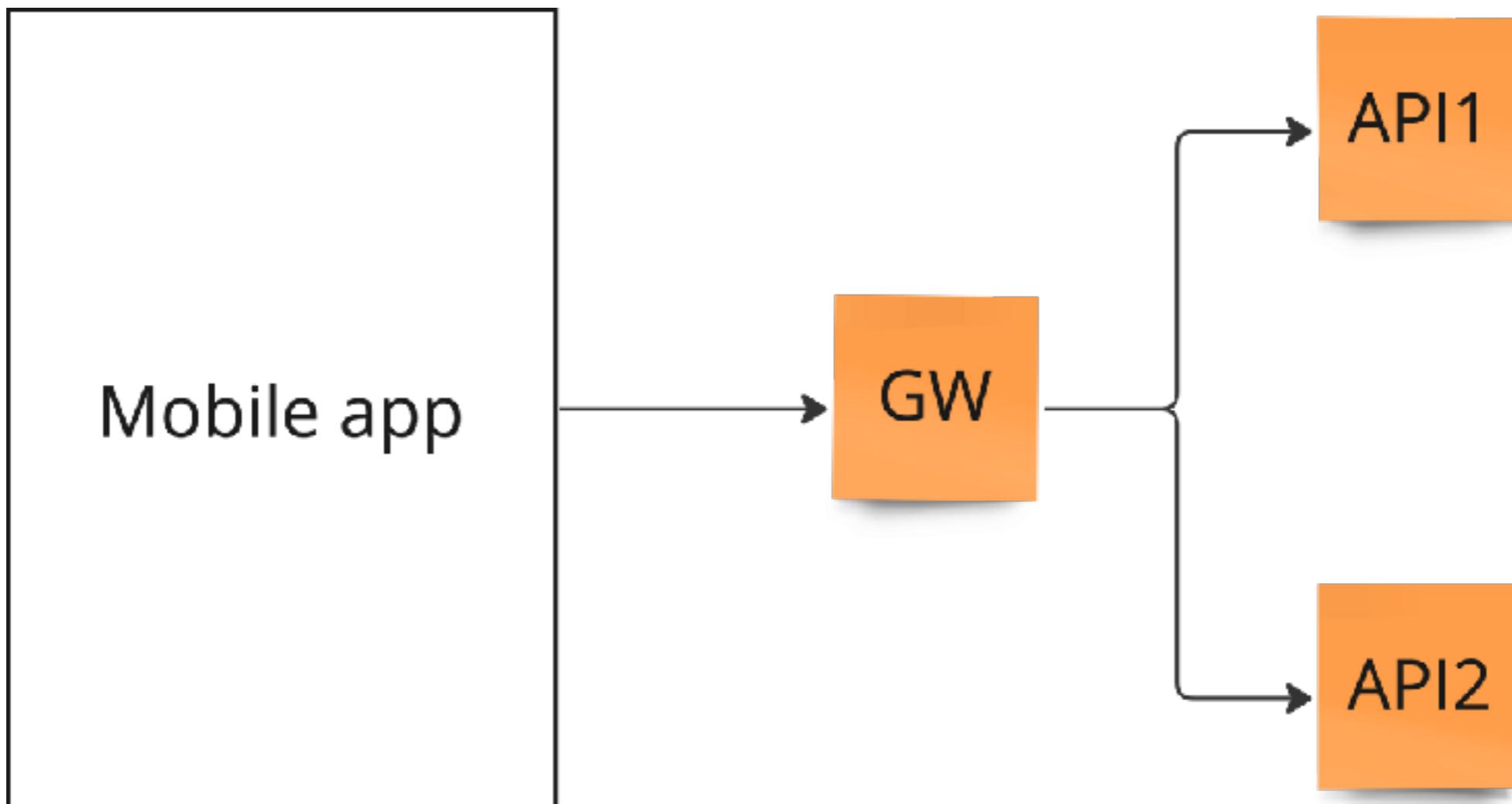
what are the technical steps required to exercise the functionality



Architecture of Mobile App ?



Architecture of System



Test strategy ?

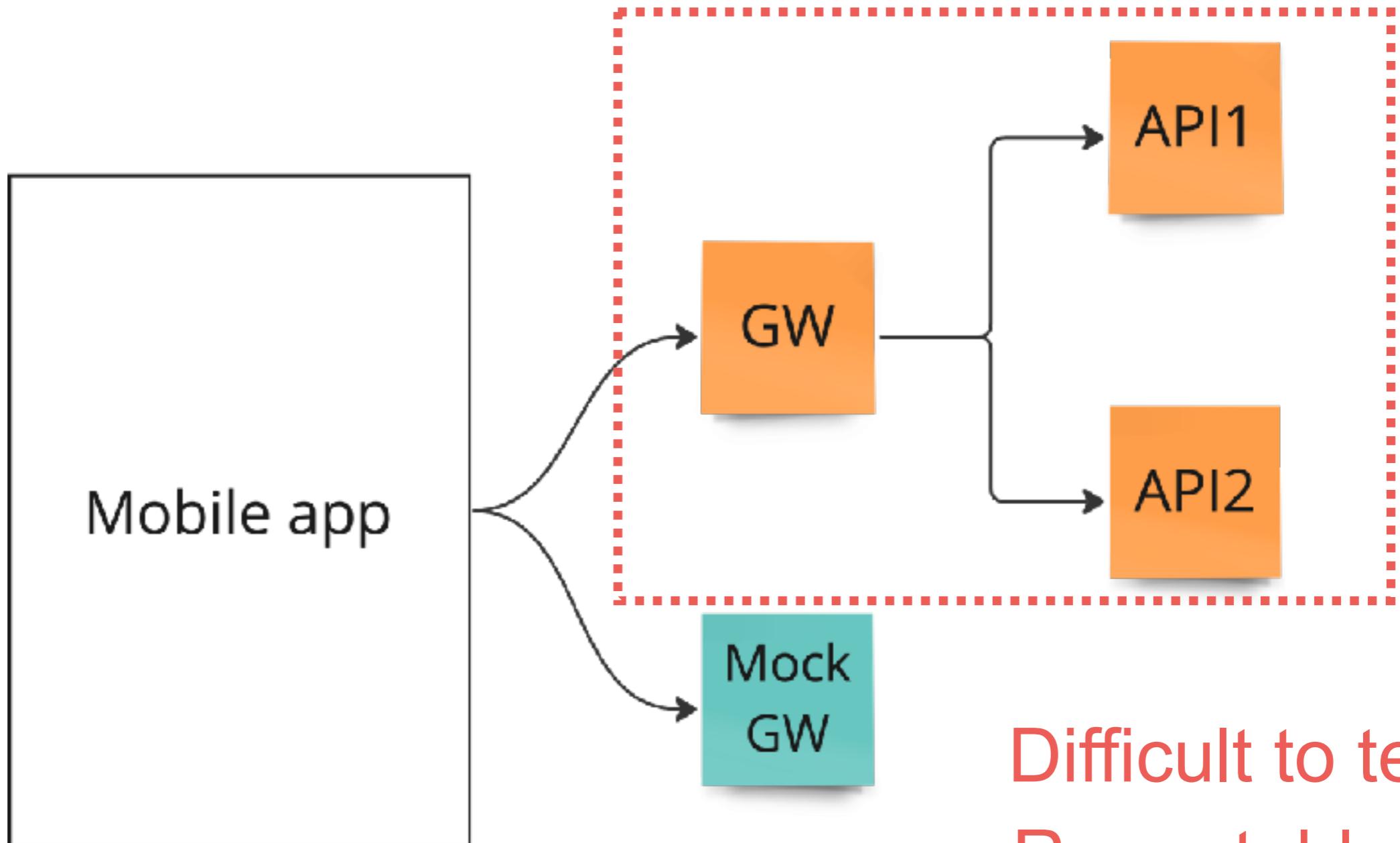


Good Tests ?

Fast
Isolate
Repeatable
Self-verify
Timely and Through
Understandable



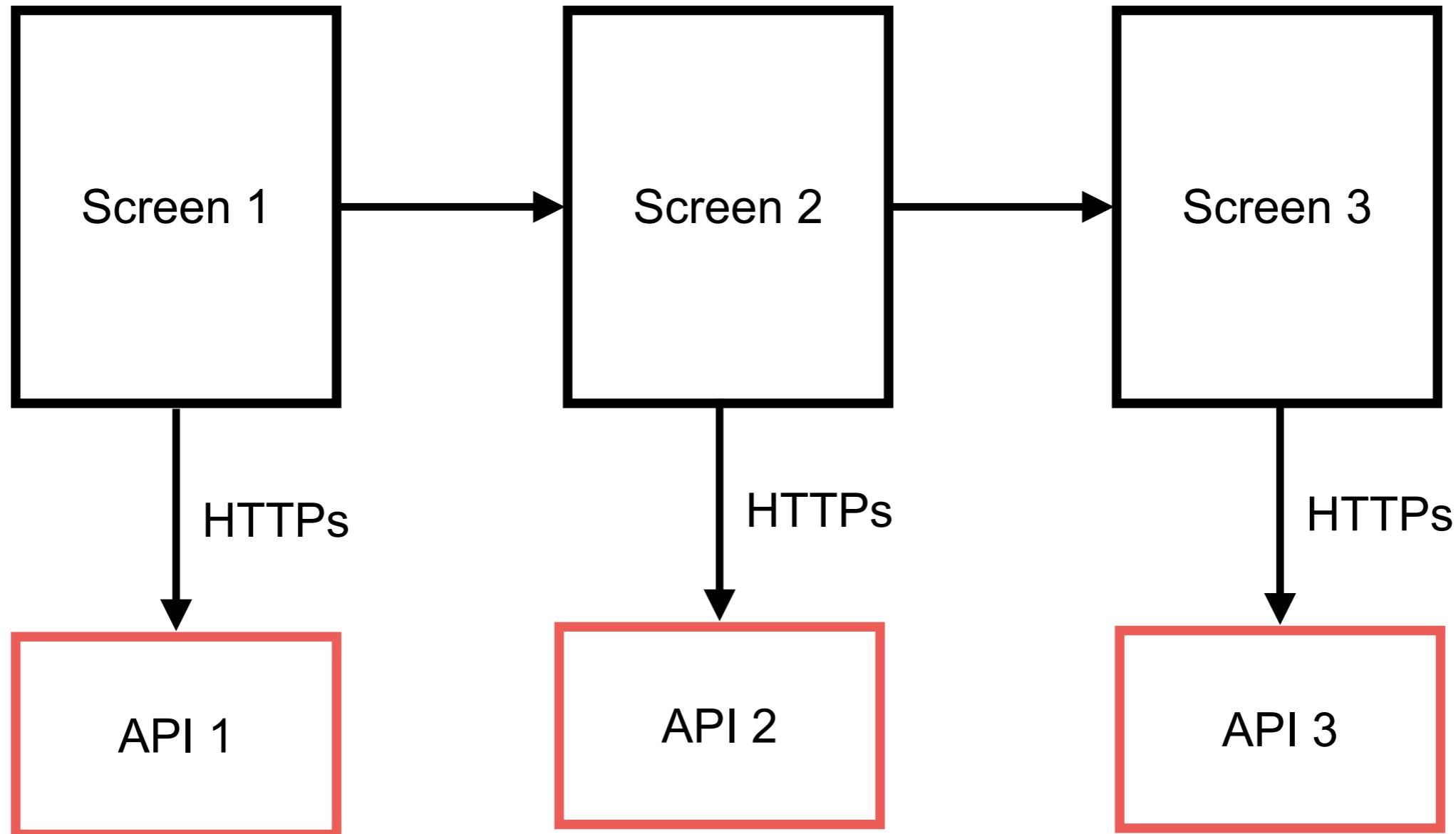
Integration Testing



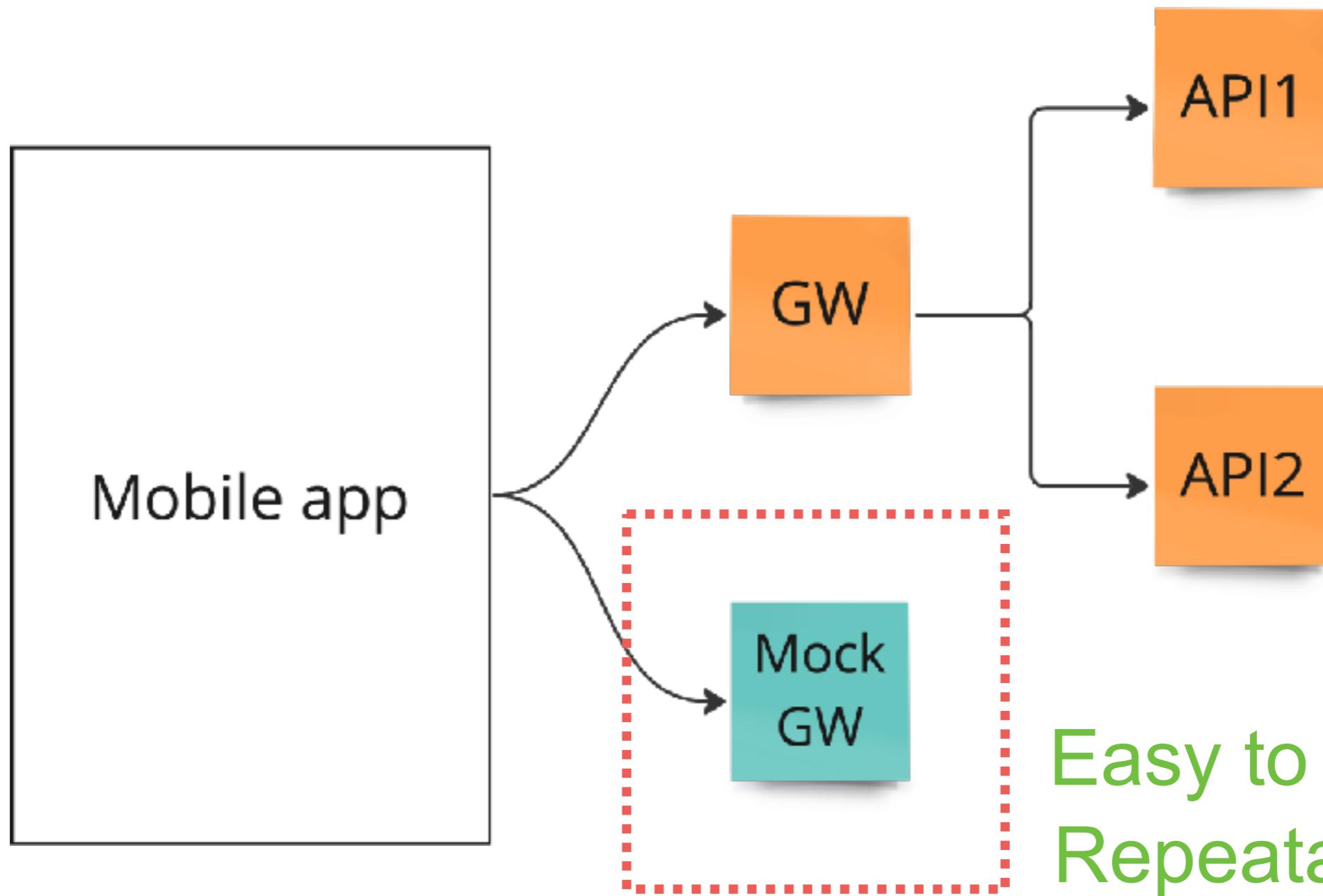
Difficult to test
Repeatable
Reliable



Integration Testing



Component Testing



Easy to test
Repeatable
Reliable



Mock Server (HTTPs)

Stubby4Node

<https://github.com/mrak/stubby4node>

MountBank

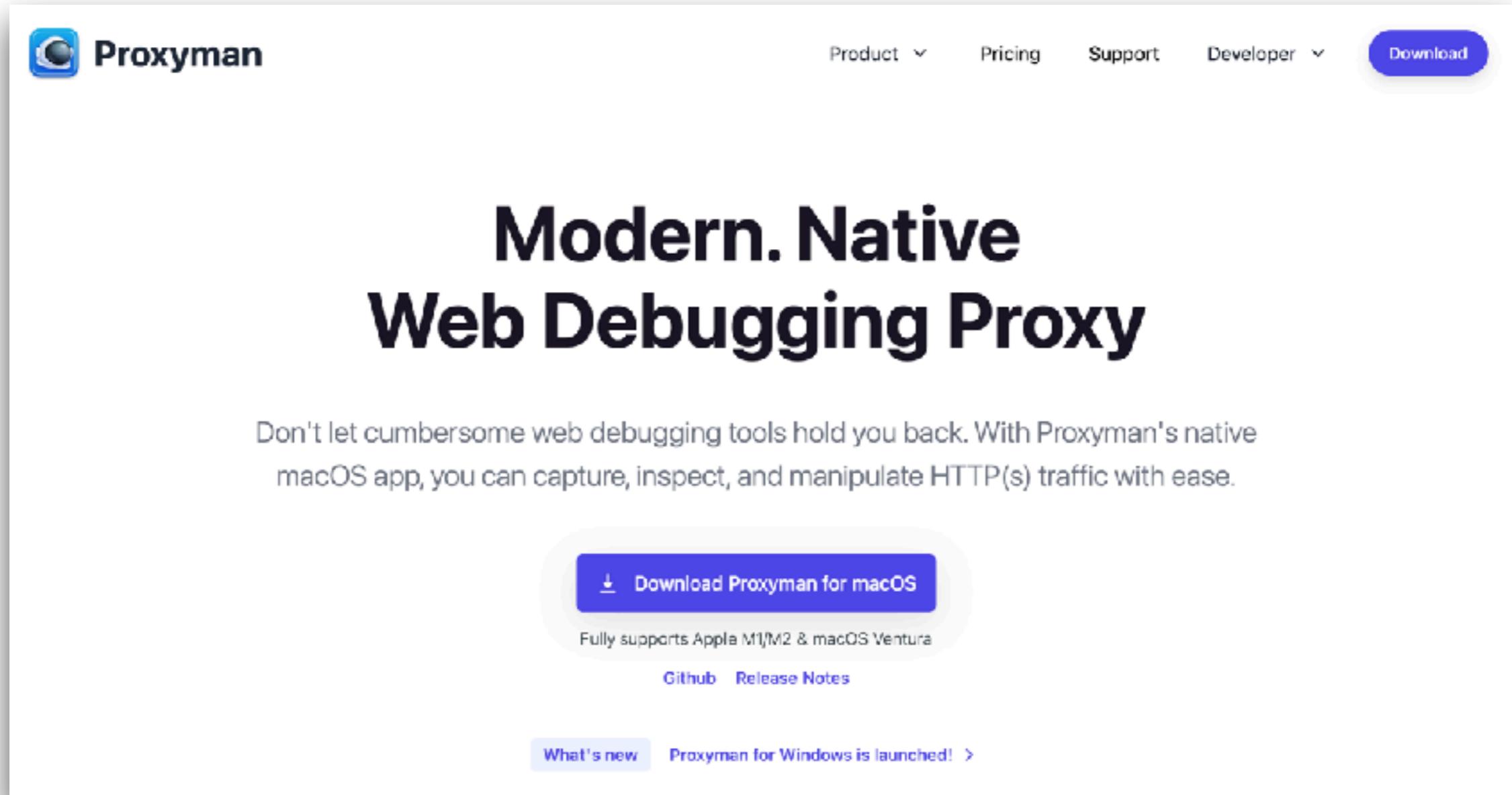
<https://www.mbttest.org/>

WireMock

<https://wiremock.org/>



Capture traffic from Mobile app

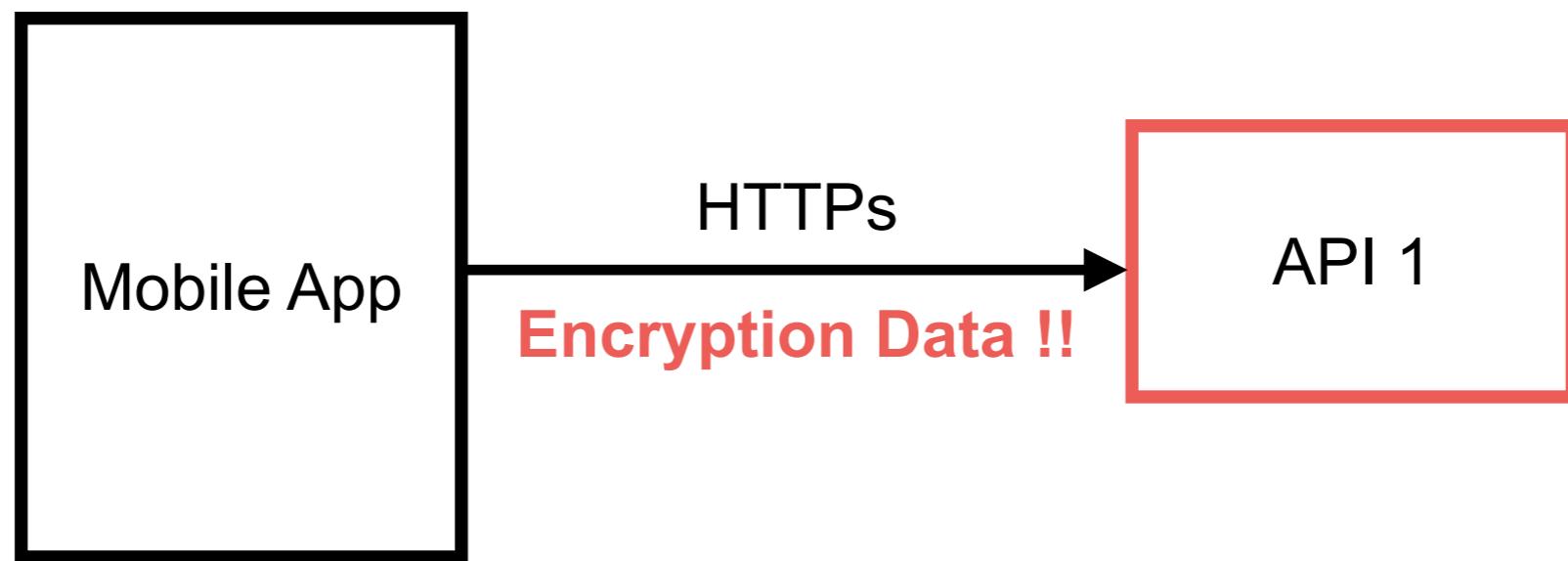


The screenshot shows the Proxyman website homepage. At the top left is the Proxyman logo (a blue square with a white gear icon). To its right are navigation links: Product (with a dropdown arrow), Pricing, Support, Developer (with a dropdown arrow), and a blue "Download" button. Below the navigation is a large, bold title: "Modern. Native Web Debugging Proxy". Underneath the title is a descriptive text: "Don't let cumbersome web debugging tools hold you back. With Proxyman's native macOS app, you can capture, inspect, and manipulate HTTP(s) traffic with ease." Below this text is a prominent blue "Download Proxyman for macOS" button with a download icon. Underneath the button, it says "Fully supports Apple M1/M2 & macOS Ventura". Below that are links to "Github" and "Release Notes". At the bottom of the main content area are two small buttons: "What's new" and "Proxyman for Windows is launched! >".

<https://proxyman.io/>



Encryption Data !!



Manage App's States

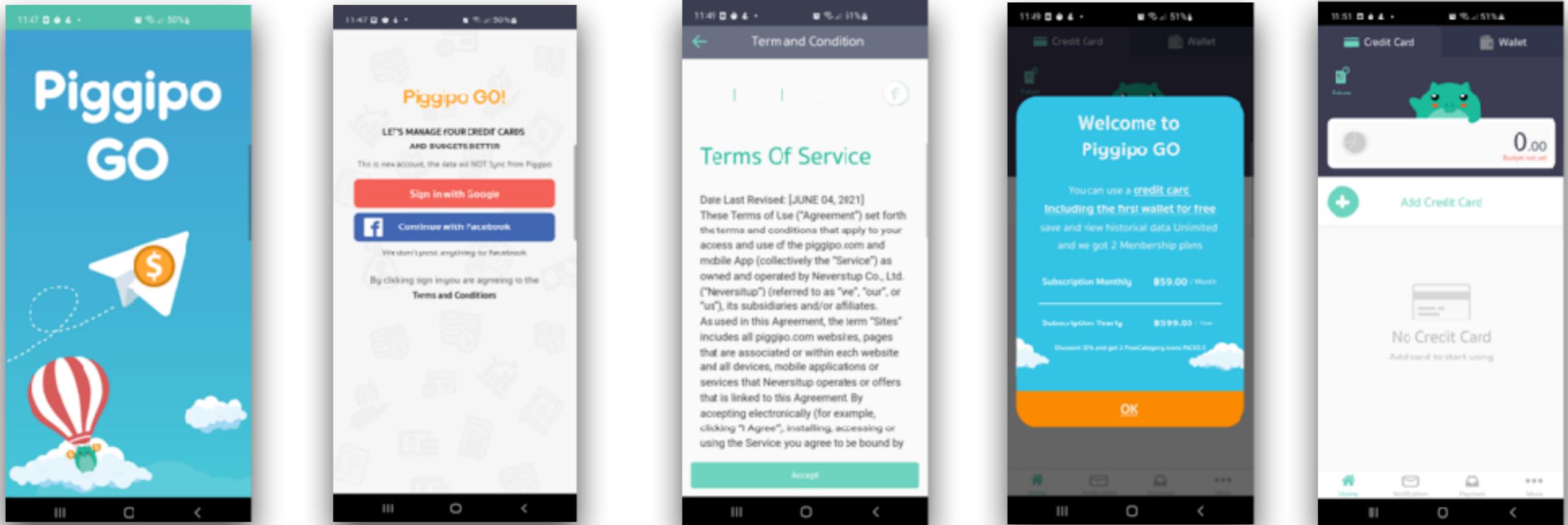


Manage App's States

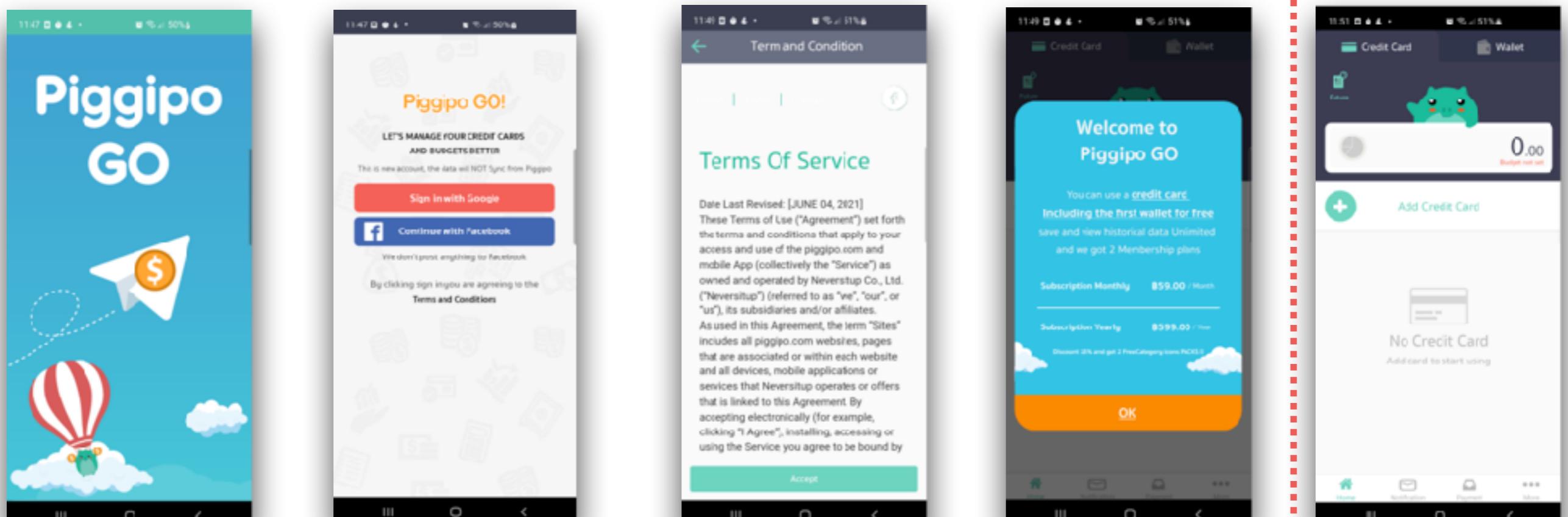
Easy for test
Reduce test steps



Manage App's States



Manage App's States



Manage App's States

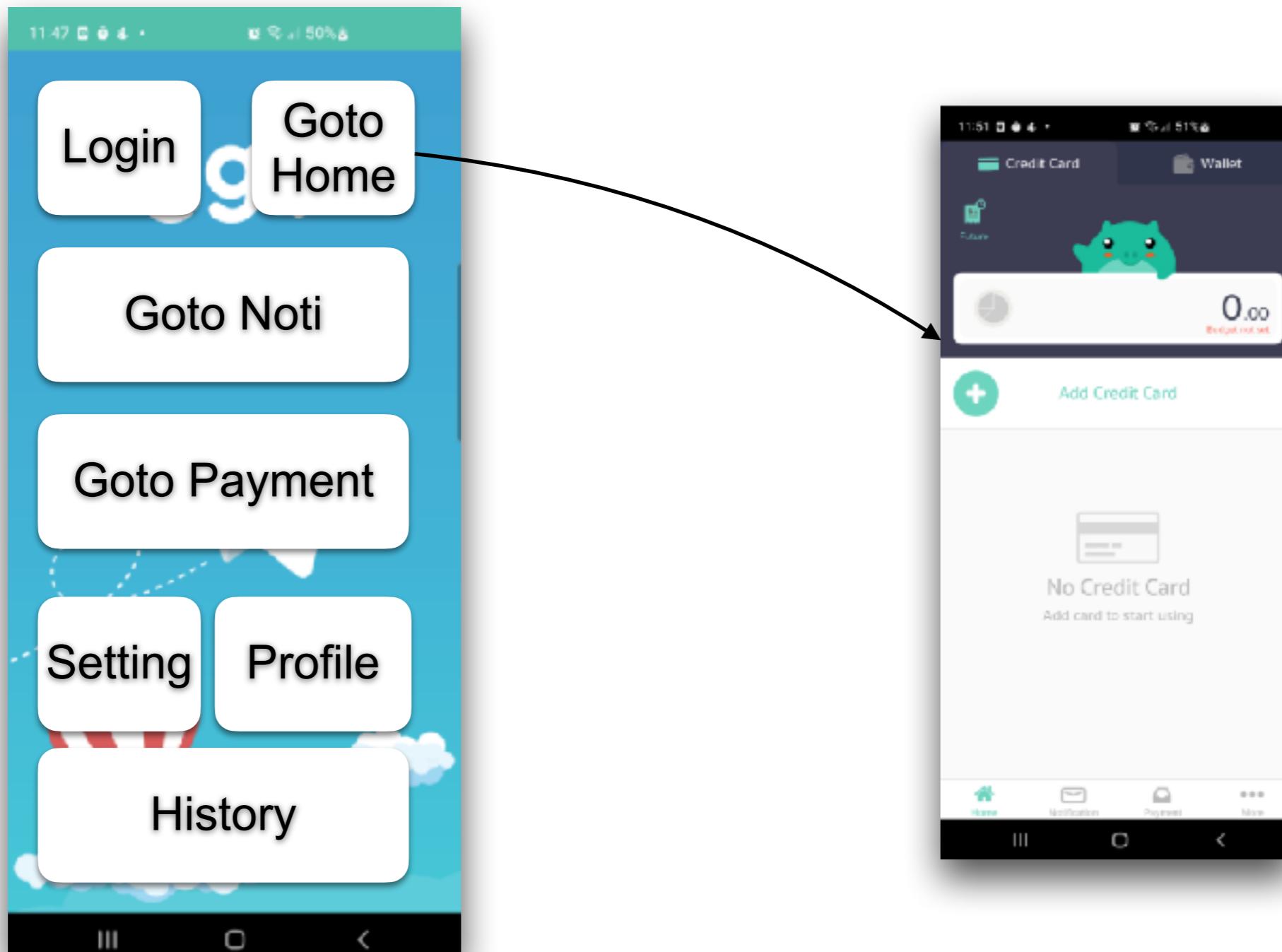
Use only in test build of the app

Screen with on-click link to all area of app

Link can even populate data



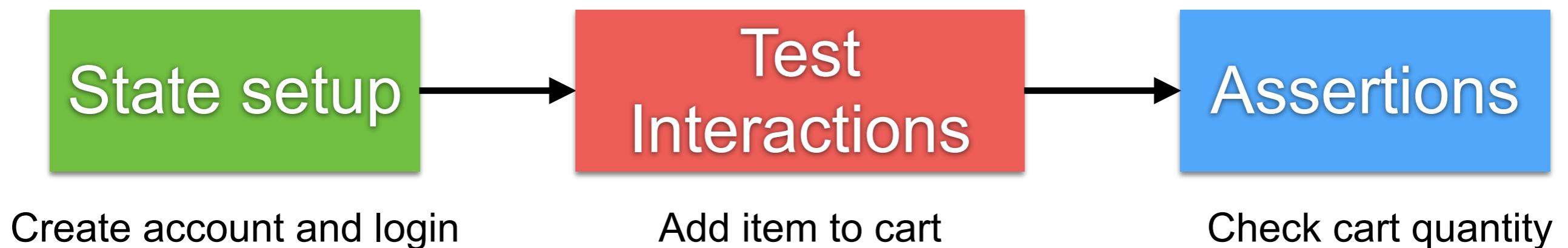
Manage App's States



State vs. Assertions

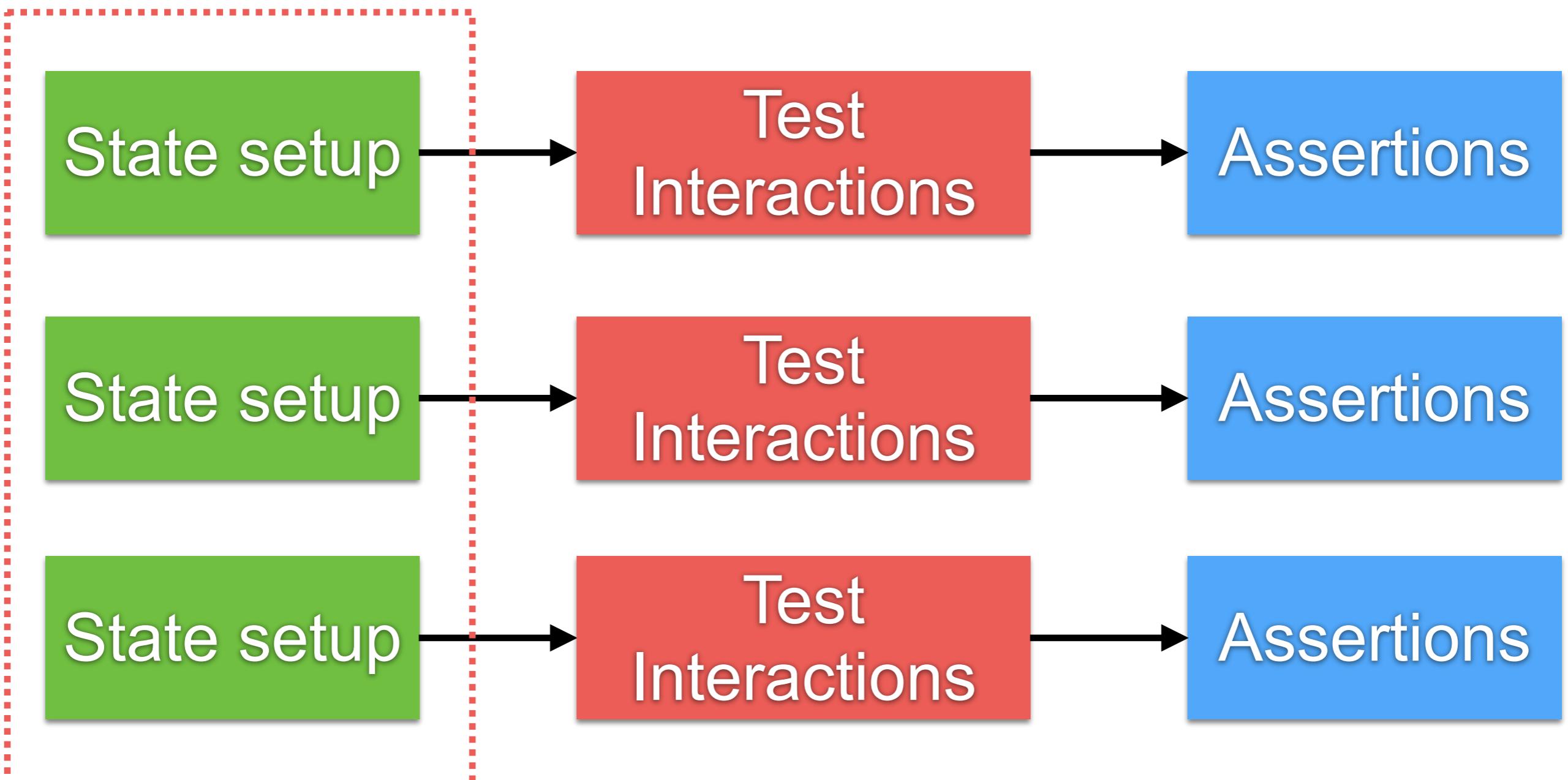


State vs. Assertions



State vs. Assertions

Not too much time !!



Make it Work



Make it Right



**Test-First
Test-Last
Test-Later**



Let's start with Appium

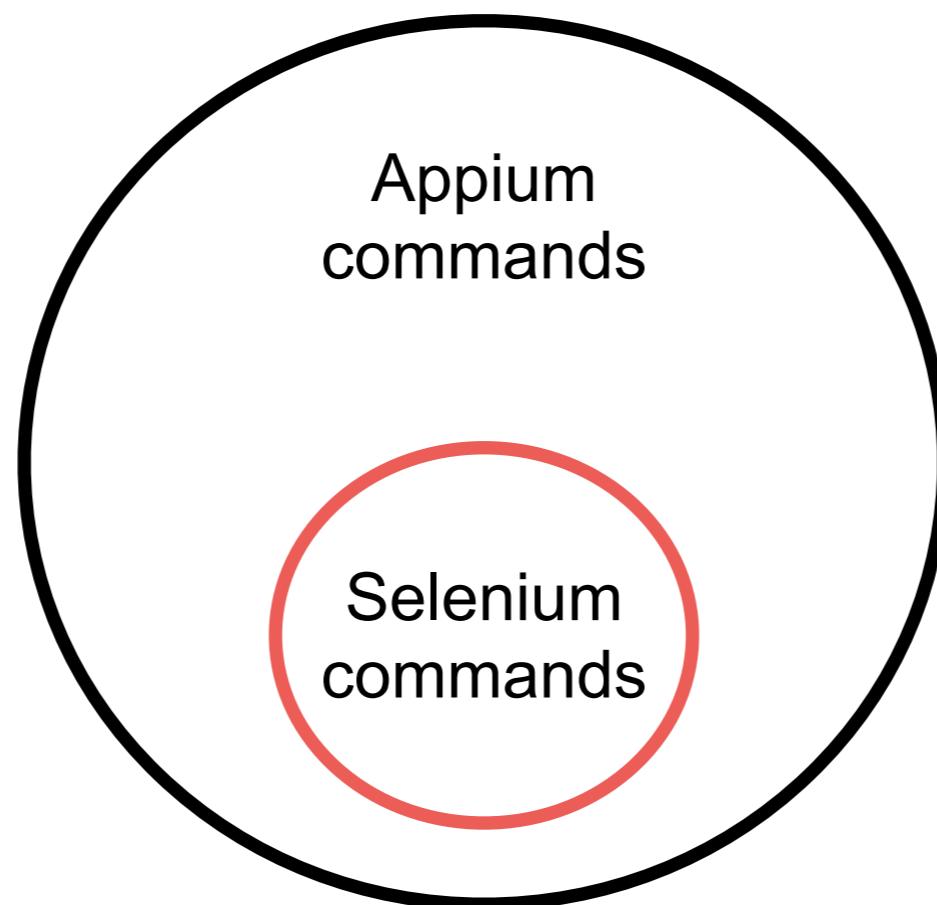


Appium

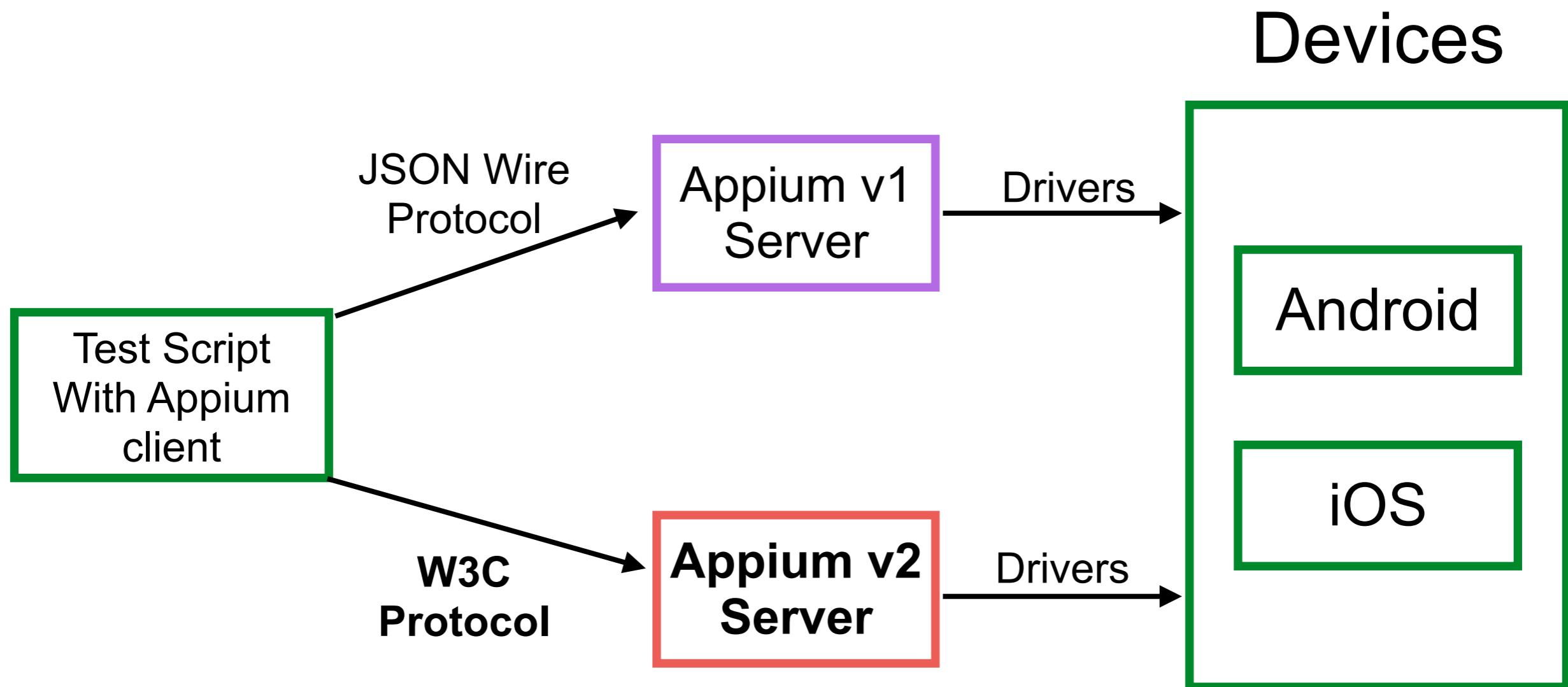
Automation tool for mobile app

Selenium for mobile

High compatibility with selenium



Architecture



Appium Tools

Appium Server via npm

Appium Server GUI/Desktop (deprecated)

Appium Inspector

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



Appium Clients

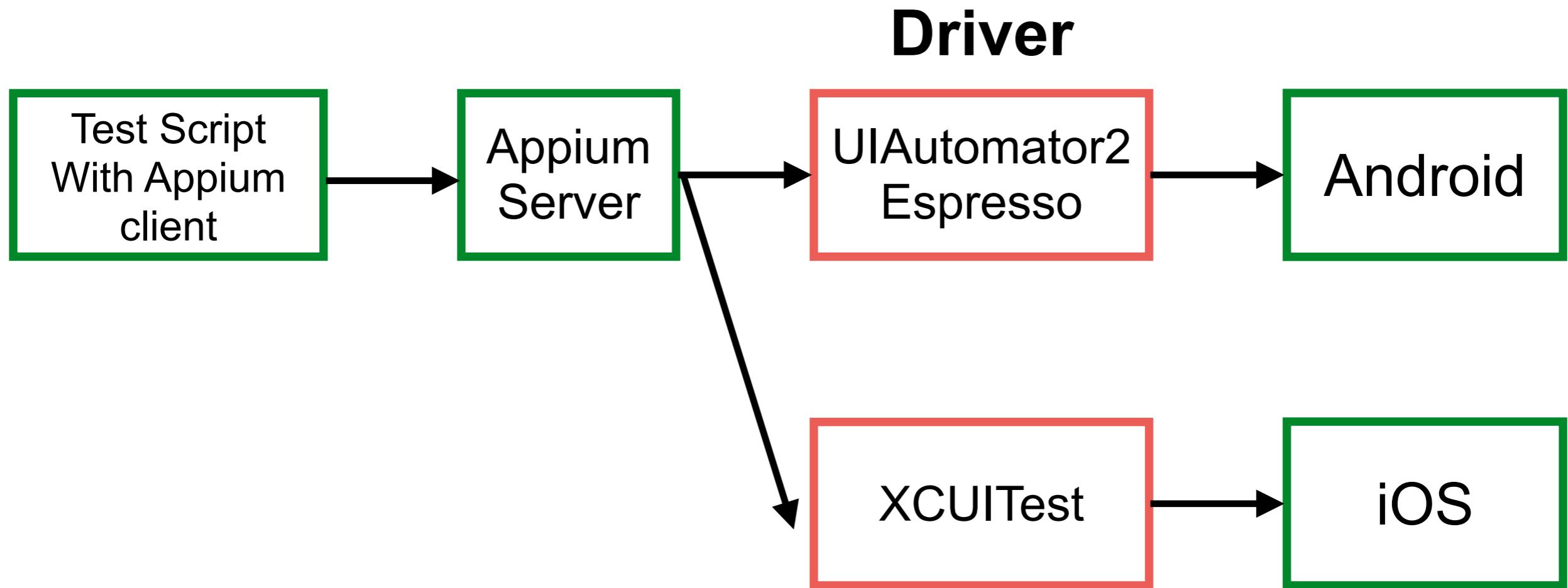
Ruby
Python
Java
C#

Robotframework + Appium Library

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



Appium Drivers and Plug-in



Appium Drivers

\$appium driver install <installation key>

Driver	Installation Key	Platform(s)	Mode(s)
Chromium	chromium	macOS, Windows, Linux	Web
Espresso	espresso	Android	Native
Gecko	gecko	macOS, Windows, Linux, Android	Web
Mac2	mac2	macOS	Native
Safari	safari	macOS, iOS	Web
UiAutomator2	uiautomator2	Android	Native, Hybrid, Web
Windows	windows	Windows	Native
XCUITest	xcuitest	iOS	Native, Hybrid, Web

<https://appium.io/docs/en/latest/ecosystem/drivers/>



Appium Plugins

Extend or modify Appium's behavior

Official Plugins

These plugins are currently maintained by the Appium team:

Plugin	Installation Key	Description
Execute Driver	execute-driver	Run entire batches of commands in a single call to the Appium server
Images	images	Image matching and comparison features
Relaxed Caps	relaxed-caps	Relax Appium's requirement for vendor prefixes on capabilities
Universal XML	universal-xml	Instead of the standard XML format for iOS and Android, use an XML definition that is the same across both platforms

<https://appium.io/docs/en/latest/ecosystem/plugins/>



Appium Interceptor Plugins

Intercept API response and mocking
Supported Android

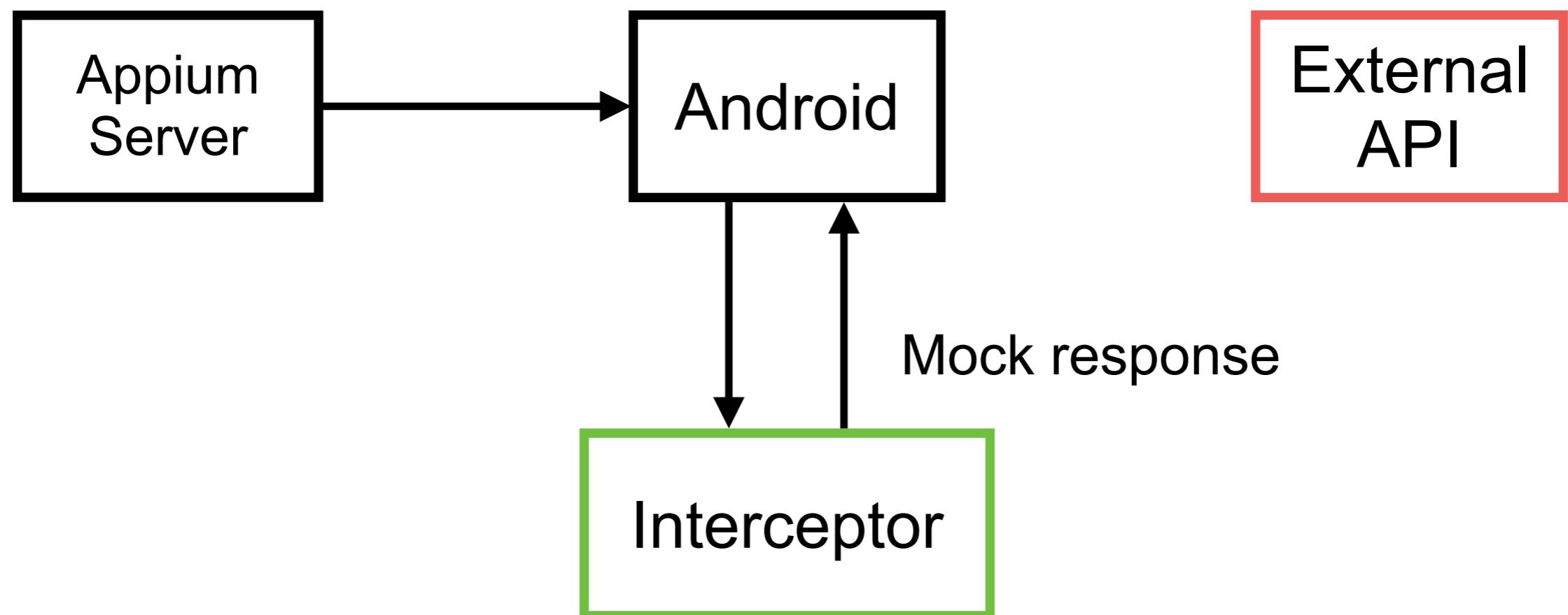


<https://github.com/Appium/TestDistribution/appium-interceptor-plugin>



Appium Interceptor Plugins

\$appium plugin install --source=npm appium-interceptor



<https://github.com/AppiumTestDistribution/appium-interceptor-plugin>



Drivers in workshop

Android

iOS

Flutter

UIAutomator2

XCUITest

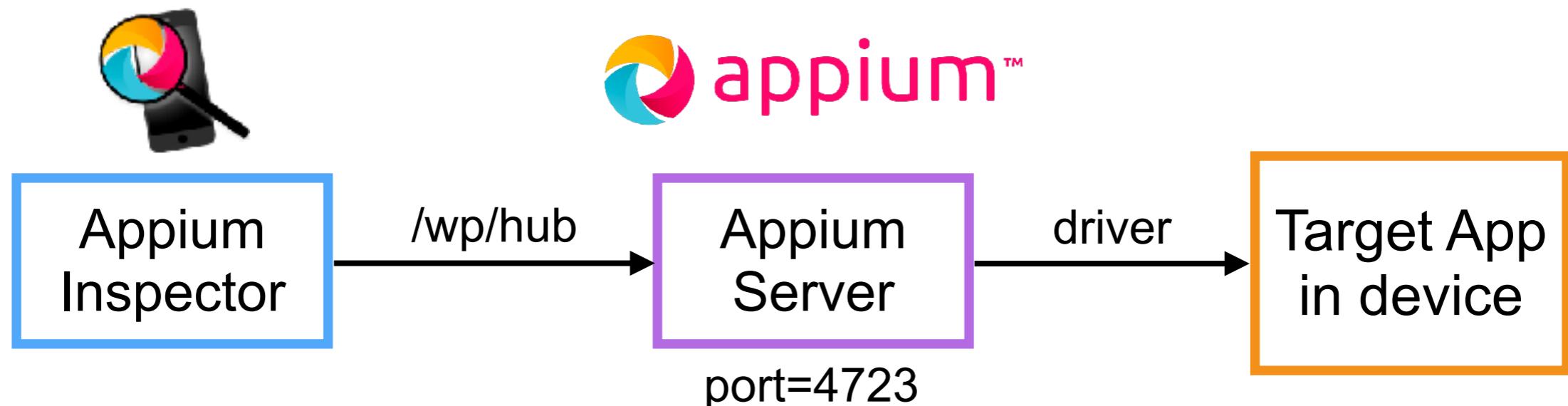
Flutter



Let's start with Appium



Let's start with Appium



Appium doctor (android, iOS)

<https://github.com/up1/course-appium-robotframework/wiki/Workshop-with-appium>



Steps to run

Appium doctor

Start server

Provide target apps (api, ipa/app)

Appium Inspector

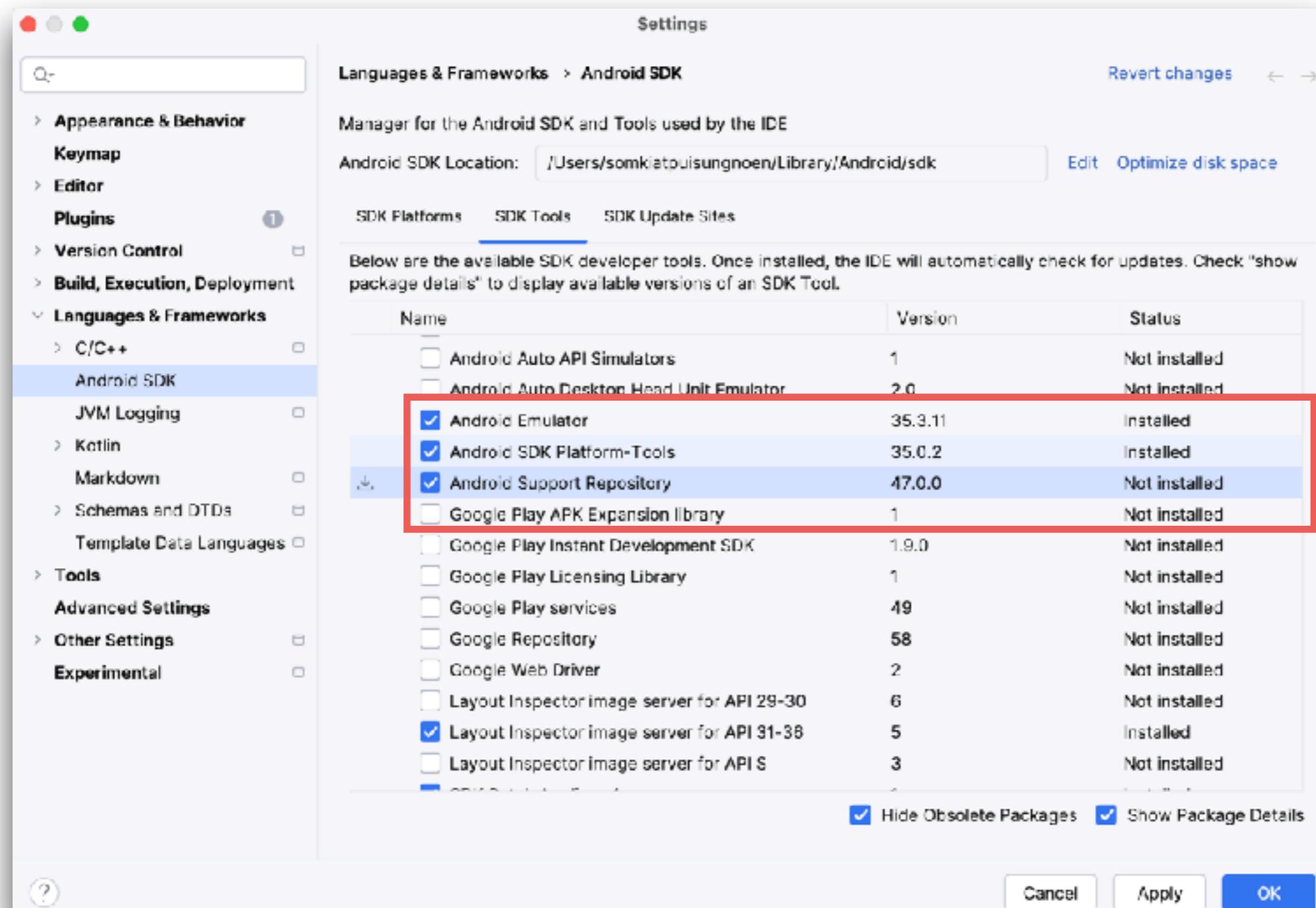


Check environments

Android	iOS
APPIUM_HOME	APPIUM_HOME
ANDROID_HOME	Xcode
JAVA_HOME	Xcode command line tools
adb, emulator, apk analyzer	ios-deploy, applesimutils
NodeJS	NodeJS



Install tools in Android Studio



List of android's devices

\$adb devices



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)
```

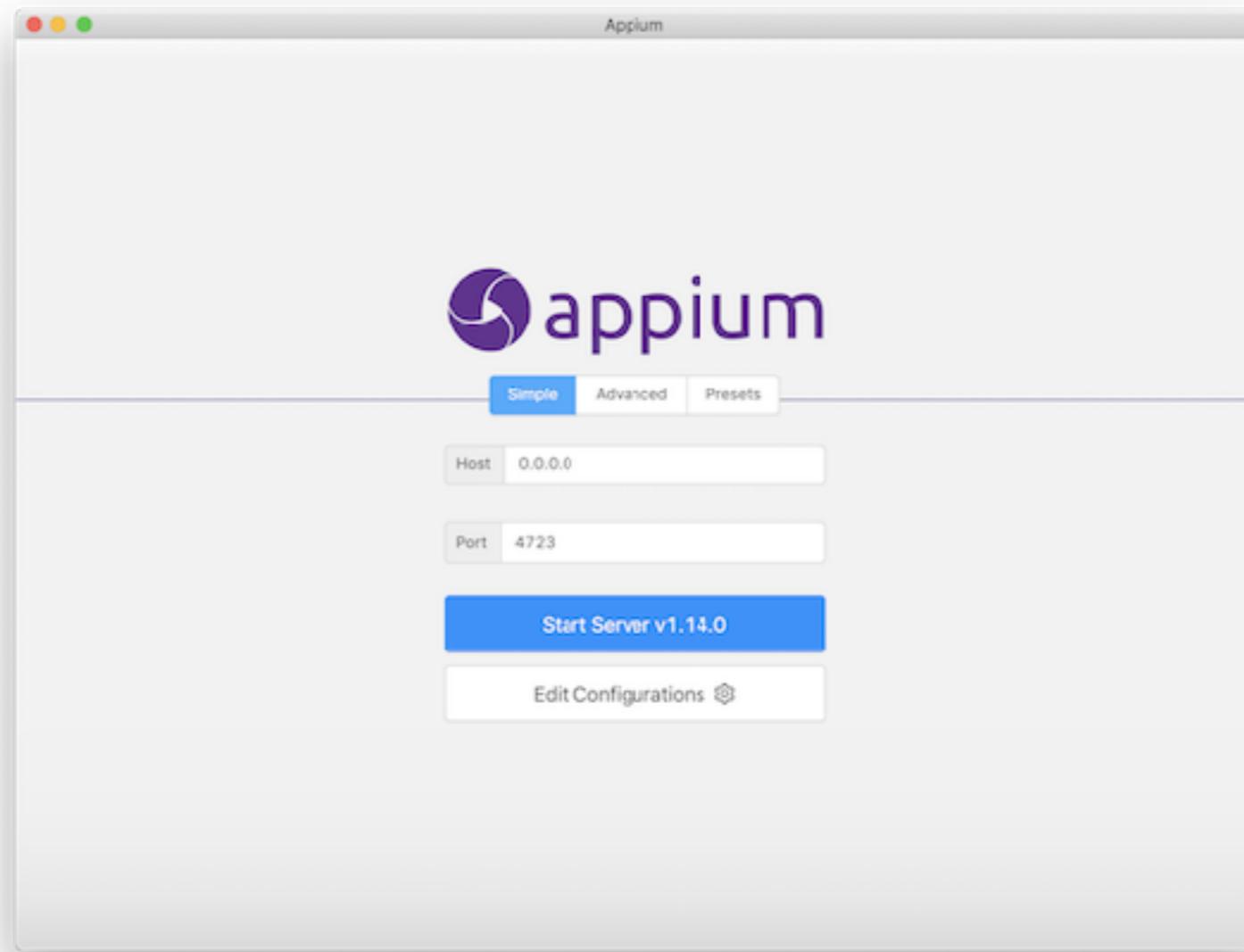


Appium Server



Appium Server Desktop

DEPRECATED !!!



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



Appium Server (CLI)

Install via npm (required Node.js)

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

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

<https://appium.io/docs/en/latest/cli/args/>

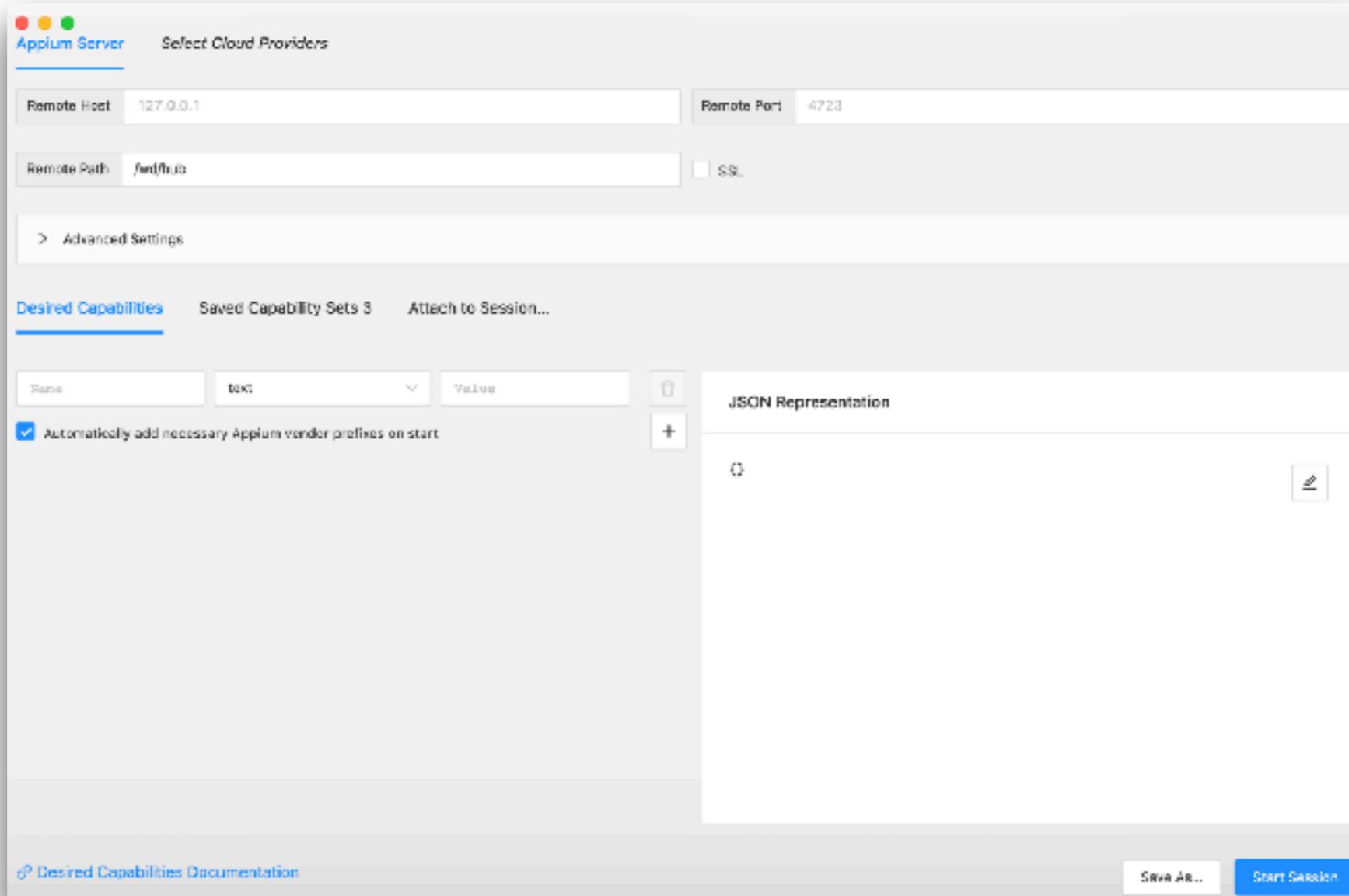


Appium Inspector



Appium Inspector

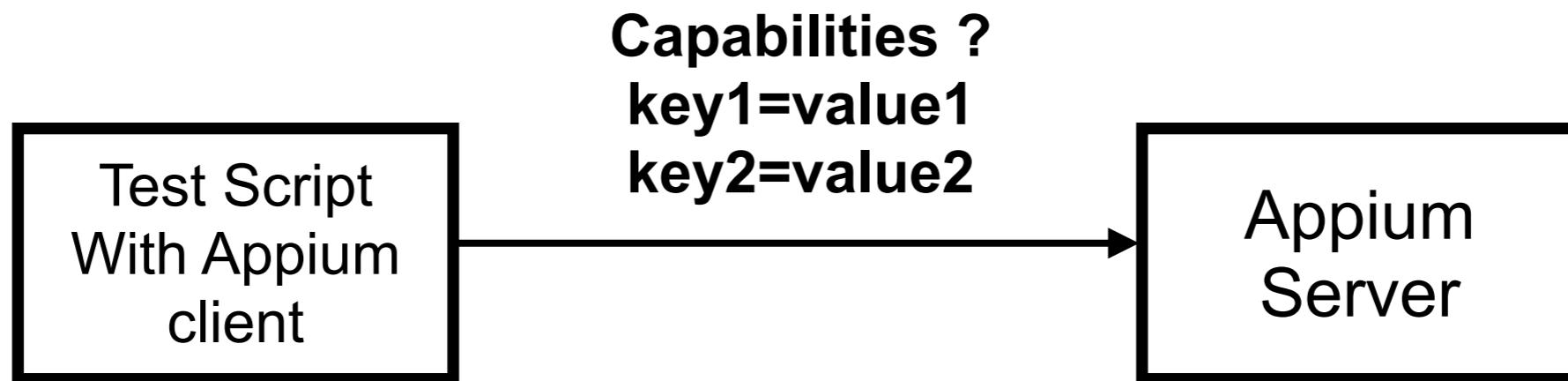
GUI inspector for mobile apps



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



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

<https://appium.io/docs/en/2.1/guides/caps/>



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/>



Appium Inspector

The screenshot shows the Appium Inspector interface with the 'Source' tab selected. On the left, a mobile application screen for 'Jetpack Compose' is displayed, featuring a logo and a 'Login' form with fields for 'Email ID or Mobile Number' and 'Password'. Below the form are links for 'Forgot Password?' and 'Login'. At the bottom, there's a link to 'Register'. On the right, the 'App Source' panel shows the XML structure of the app's UI components, starting with a FrameLayout and moving down through LinearLayout, FrameLayout, ComposeView, View, ScrollView, and TextView elements. The 'Selected Element' panel on the far right details the properties of the currently selected 'Email ID or Mobile Number' input field, including its selector, attributes like elementId and package, and its current state (text, checked, clickable, enabled).

Selected Element

- Interactions for this element may not be available

Find By Selector

-android uiautomator (does) new UiSelector().text("Email ID or I")

xpath //android.widget.TextView[@text="Email ID or Mobile Number"]

Attribute	Value
elementId	0000000-0000-0026-0000-001600000000
index	0
package	com.deme.mylogin
class	android.widget.TextView
text	Email ID or Mobile Number
checkable	false
checked	false
clickable	false
enabled	true



Recording

The screenshot shows a mobile application interface for 'Jetpack Compose'. The top bar displays the title 'Jetpack Compose'. The main screen is titled 'Login'. It contains an input field labeled 'Email ID or Mobile Number', a password input field with a visibility icon, and a large blue 'Login' button. Below the button is a 'Forgot Password?' link. At the bottom, there is a link to 'Register'.

The 'Source' tab is selected in the recording tool, showing the XML structure of the UI components:

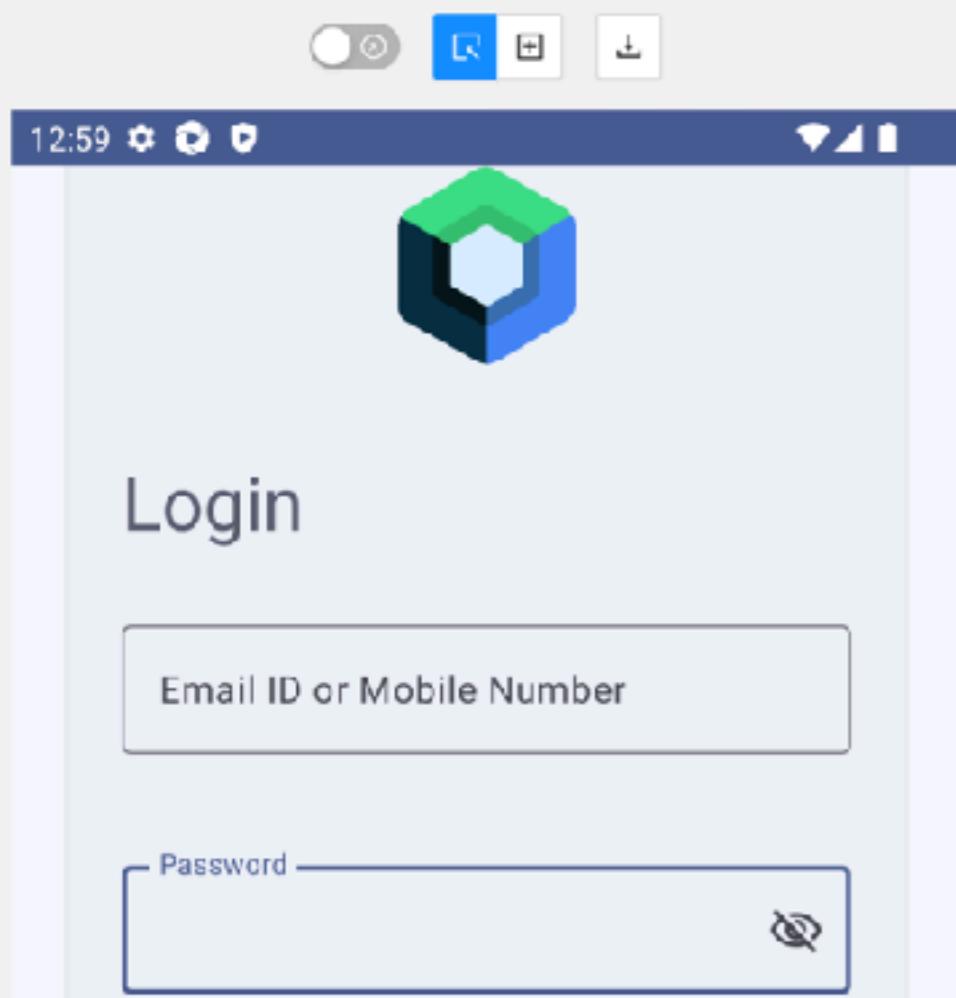
```
<android.widget.FrameLayout>
    <android.widget.LinearLayout>
        <android.widget.FrameLayout resource-id="android:id/content">
            <androidx.compose.ui.platform.ComposeView>
                <android.view.View>
                    <android.view.View>
                        <android.widget.ScrollView>
                            <android.view.View>
                                <android.widget.TextView>
                                    <string>Compose</string>
                                <android.widget.ImageView desc="Login" />
                                <android.widget.TextView>
                                    <string>Email ID or Mobile Number</string>
                                <android.widget.EditText>
                                    <string>Email ID or Mobile Number</string>
                                <android.widget.TextView desc="Password?" />
                                <android.view.View>
                                    <android.widget.TextView to=""/>
                                <android.widget.TextView to=""/>
                            </android.view.View>
                        </android.widget.ScrollView>
                    </android.view.View>
                </androidx.compose.ui.platform.ComposeView>
            </android.widget.FrameLayout>
        </android.widget.LinearLayout>
    </android.widget.FrameLayout>
```

The 'Selected Element' panel shows the details of the currently selected element, which is the input field for 'Email ID or Mobile Number'. The element has the following attributes:

Attribute	Value
elementId	00000000-0000-0026-0000-001600000000
index	0
package	com.deme.mylogin
class	android.widget.TextView
text	Email ID or Mobile Number
checkable	false
checked	false
clickable	false
enabled	true



Generate Test Script



The screenshot shows a mobile application interface for a login screen. At the top, there is a navigation bar with icons for settings, search, and download. Below the navigation bar, the title "Recorder" is displayed in blue, indicating the current mode. The main content area shows the recorded test script:

```
$el1 = Set Variable    android=new UiSelector().resourceId("testTagLoginName")
Click Element ${el1}
$el2 = Set Variable    android=new UiSelector().resourceId("testTagLoginPassword")
Click Element ${el2}
```

On the right side, there is a sidebar titled "Robot Framework" which lists other framework options: .NET - NUnit, JS - Webdriver.io, JS - Oxygen HQ, Java - JUnit4, Java - JUnit5, Python, Ruby, and Robot Framework (which is currently selected).



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>

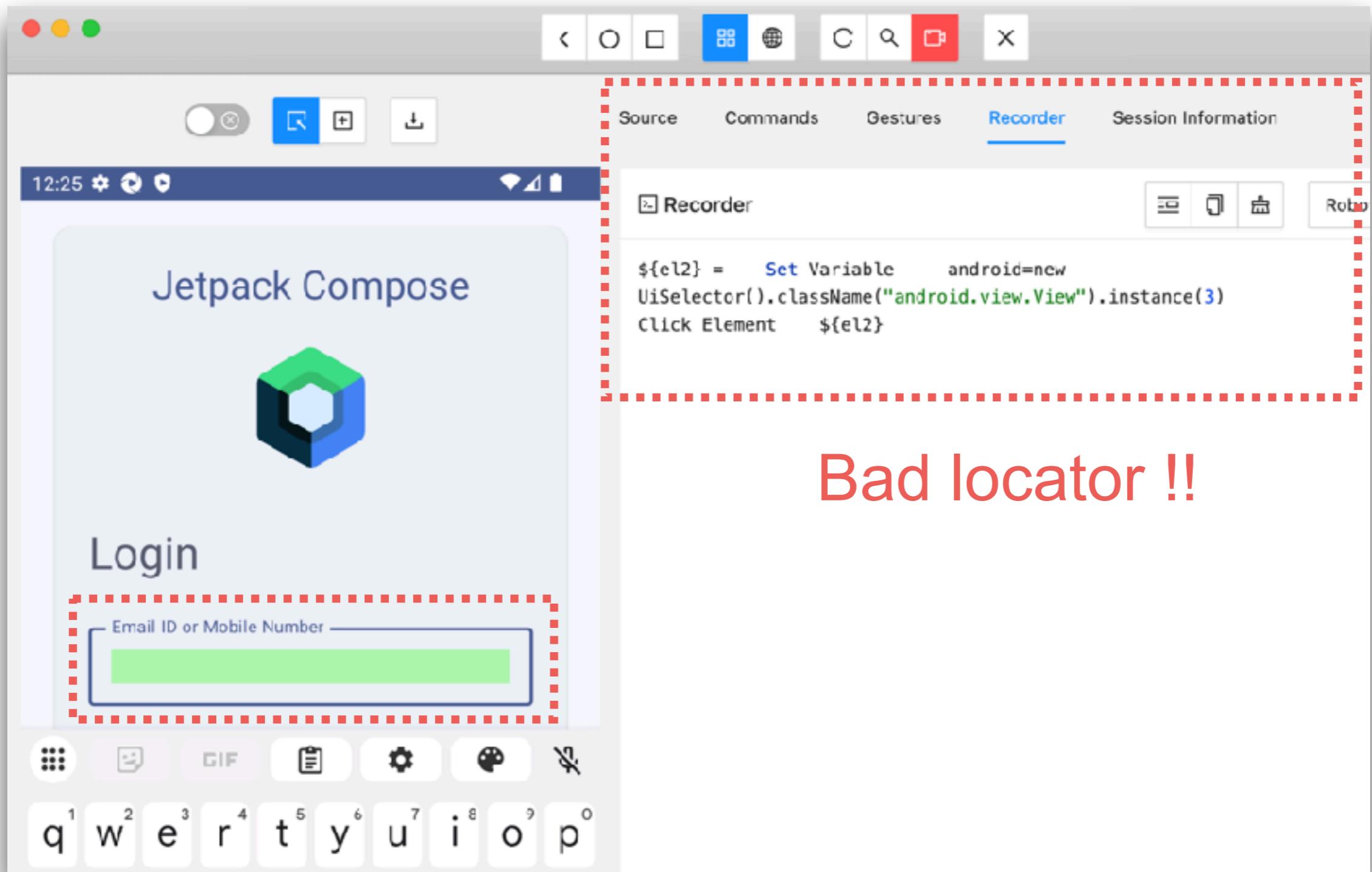


Finding and Using Elements

Isolated from UI !!



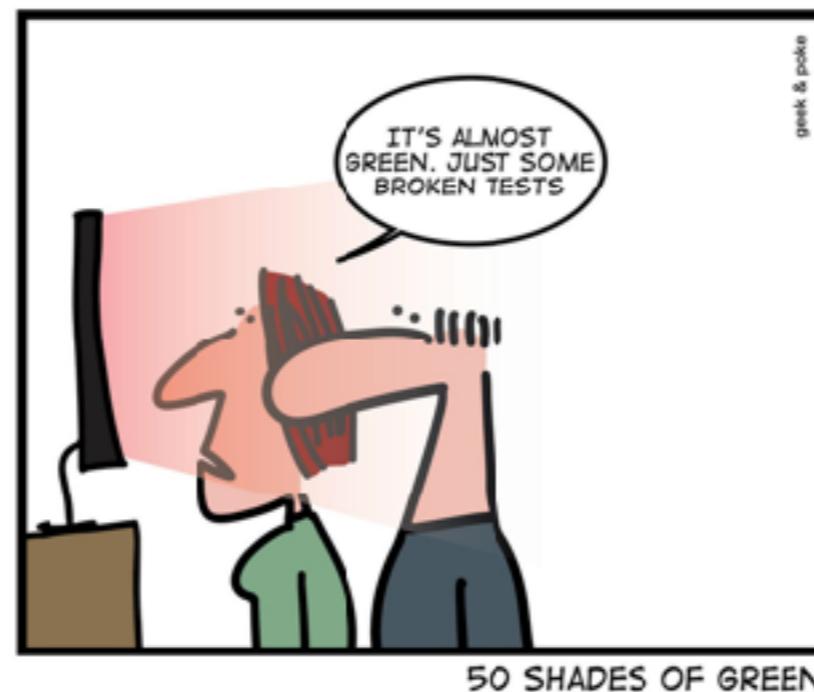
Recording and Inspect element



Good Locators

Unique
Descriptive
Resilient

Shorter in length (maintainability)



Locator Strategies for Android

Strategy	Description
ID	resource-id
Accessibility ID	Content description
Class name	UiSelector()
Android UIAutomator	Name of element
XPath	Pattern of element in XML (not recommended) Absolute path vs Relative path ?

<https://github.com/appium/appium-uiautomator2-driver/blob/master/README.md#element-location>



Improve locator in Android app

Jetpack Compose with `testTag`

Enable interoperability

```
modifier = modifier.semantics {  
    testTagsAsResourceId = true  
}
```

Set `testTag` in each element

```
EmailTextField(  
    modifier = Modifier  
        .testTag("testTagLoginName")
```

<https://developer.android.com/develop/ui/compose/testing/interoperability>



Improve locator in Android app

Jetpack Compose with **contentDescription**

```
EmailTextField(  
    modifier = Modifier  
        .fillMaxWidth()  
        .padding(top = AppTheme.dimens.paddingLarge)  
        .testTag("testTagLoginName")  
  
        .semantics {  
            contentDescription = "login_name"  
        }  
)
```

<https://developer.android.com/develop/ui/compose/testing/interoperability>



Appium Inspector

The screenshot shows the Appium Inspector interface with the 'Source' tab selected. On the left, a mobile application screen for a 'Jetpack Compose' login app is displayed. The screen features a green hexagonal logo at the top, followed by a 'Login' title. Below it are two input fields: a green one labeled 'Email ID or Mobile Number' and a white one labeled 'Password'. A 'Forgot Password?' link and a blue 'Login' button are also present. The entire application area is highlighted with a red dashed border. On the right, the 'App Source' panel shows the XML structure of the UI components. A specific element, an 'EditText' with resource ID 'testTagLoginName', is highlighted with a blue selection bar and is shown in the 'Selected Element' panel. This panel displays the element's selector ('id: testTagLoginName'), its class ('android.widget.EditText'), and its attributes, including 'text' (empty), 'resource-id' ('testTagLoginName'), and 'checked' ('false'). Other attributes like 'index', 'package', 'class', 'clickable', 'enabled', 'focusable', and 'focused' are also listed.

Source Commands Gestures Recorder Session Information

12:48

App Source

Selected Element

Enter Keys to Send

Find By Selector

id testTagLoginName

-android uiautomator (docs) new UiSelector().resourceId("testTagLoginName")

xpath //android.widget.EditText[@resource-id="testTagLoginName"]

Attribute	Value
elementId	0000000-0000-0065-0000-00000004
index	3
package	com.demo.mylogin
class	android.widget.EditText
text	
resource-id	testTagLoginName
checked	false
checked	false
clickable	true
enabled	true
focusable	true
focused	false



XPath

XML Path Language

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

XPath is powerful but very dangerous



Benefits of XPath

Find any element that exists

Find elements by using complex criteria



Cons of XPath

XPath selectors can be brittle

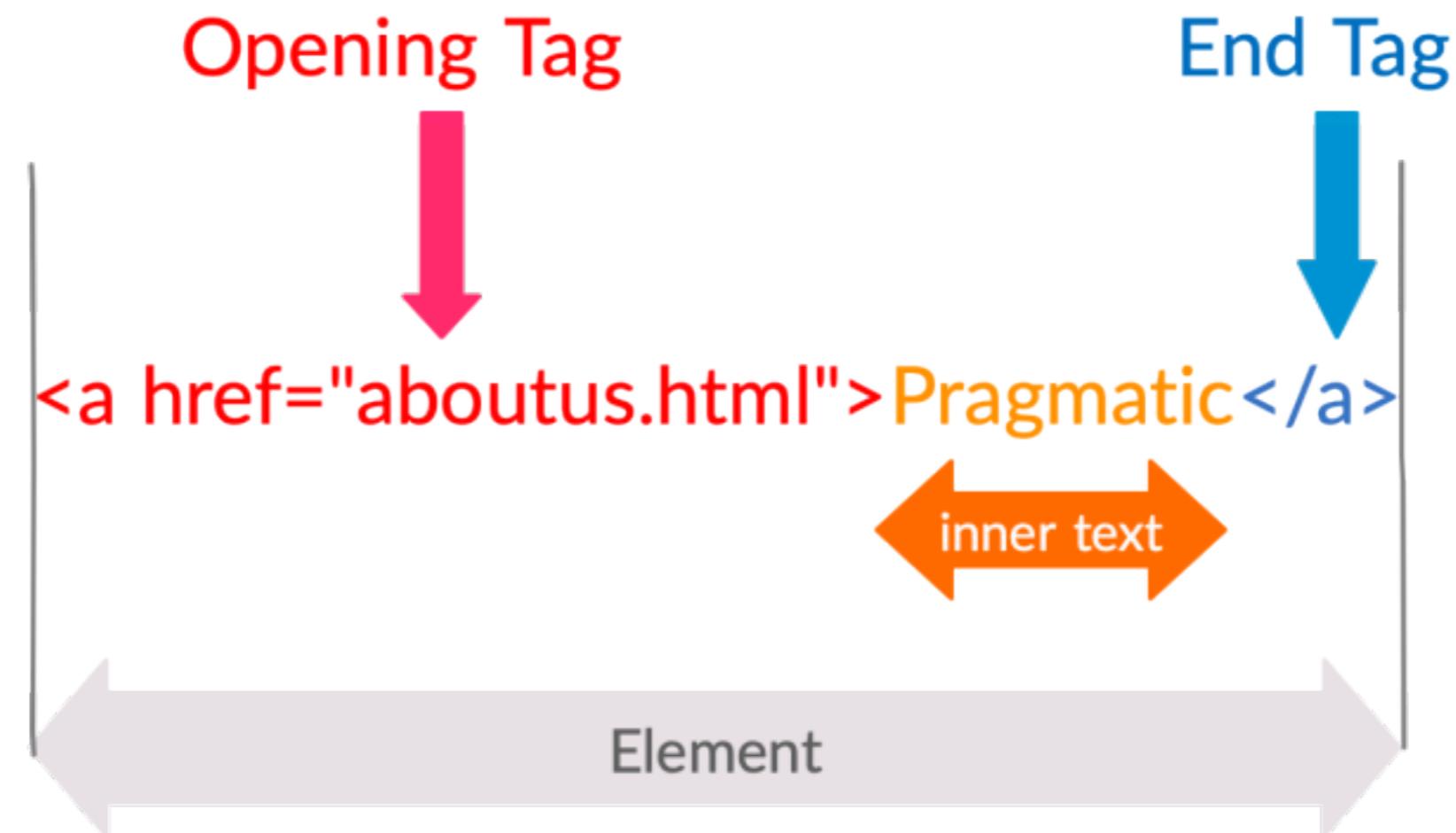
Slow

Broken test !!

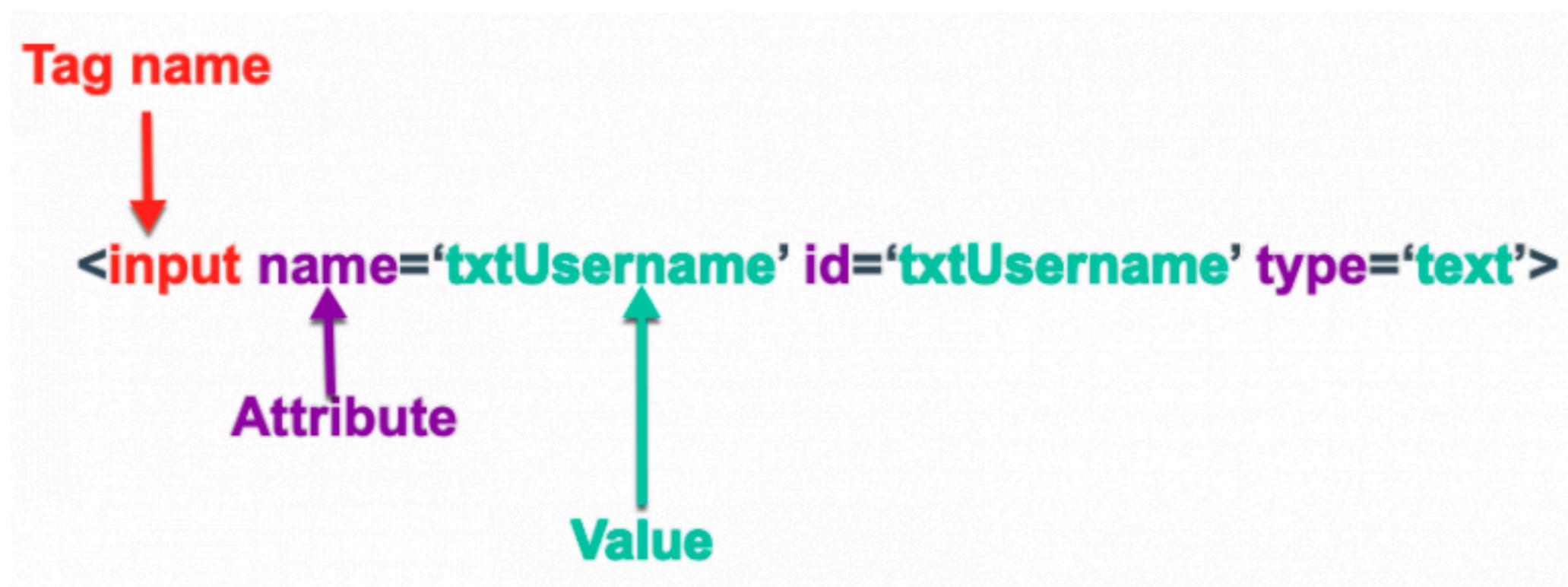
Avoid brittle selectors by rely on unique tag attribute



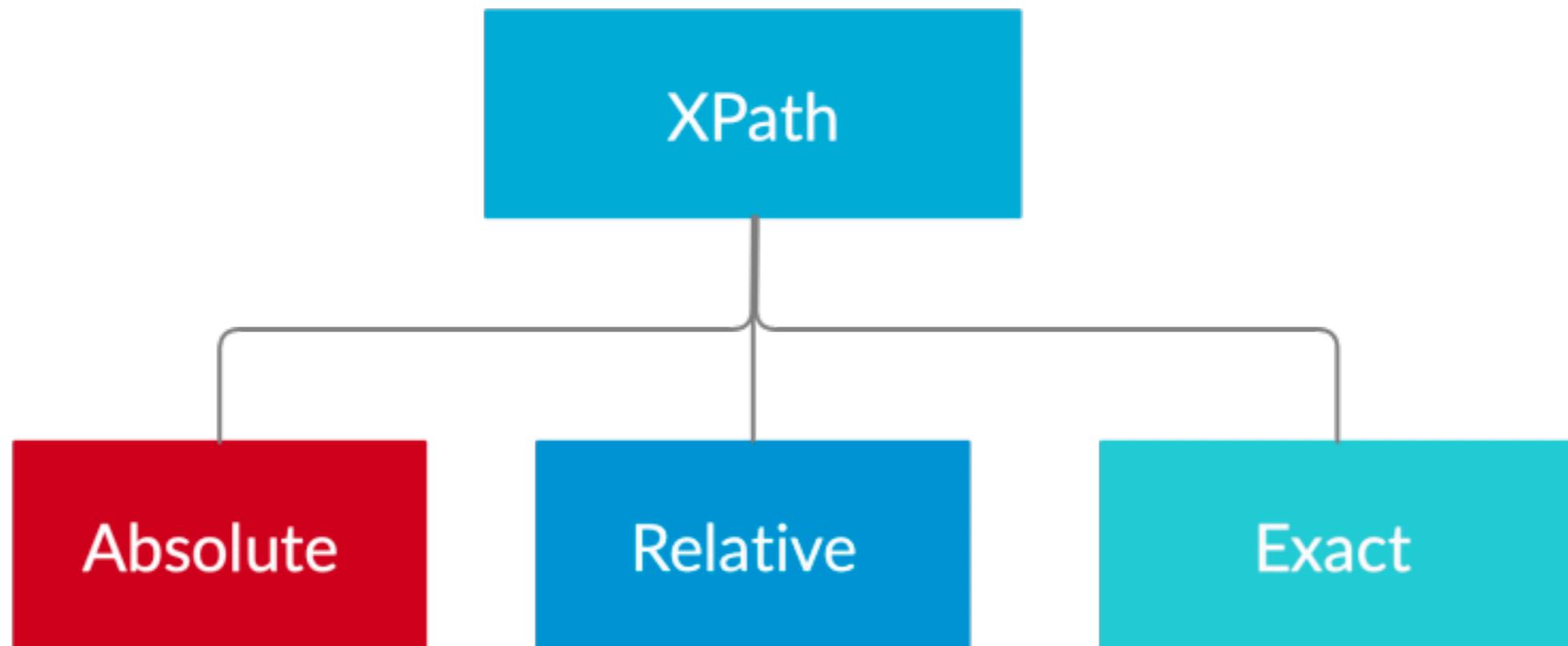
XPath



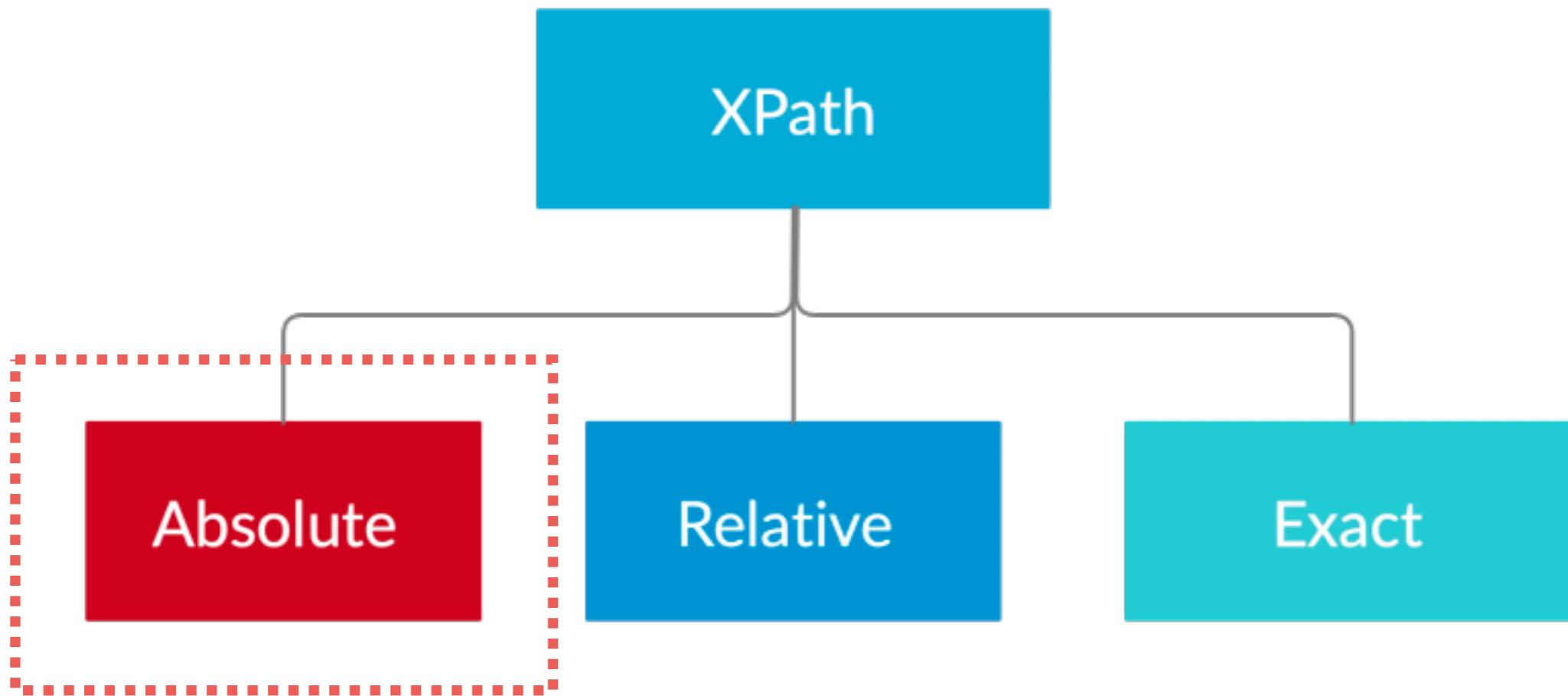
XPath



Types XPath



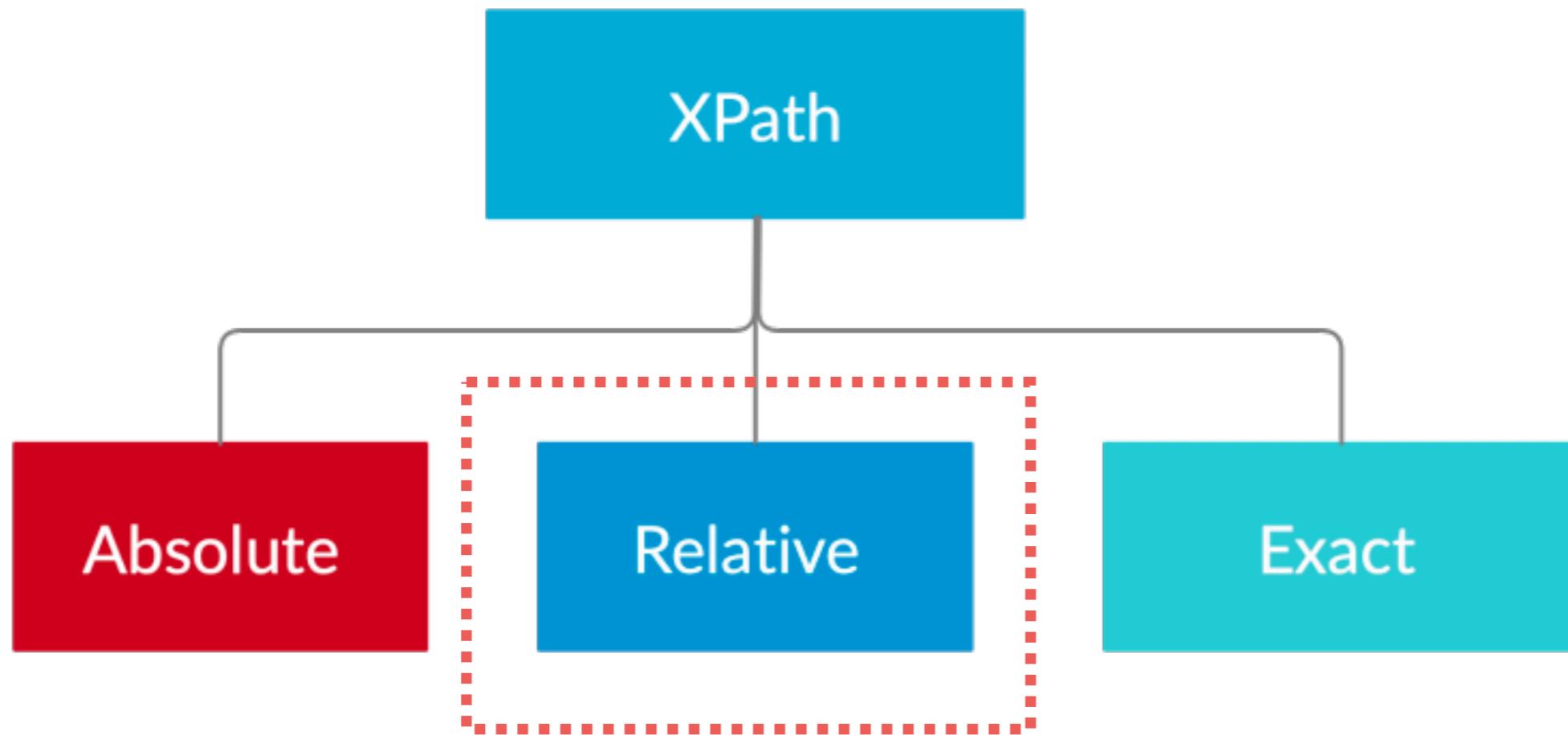
Don't use !!



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



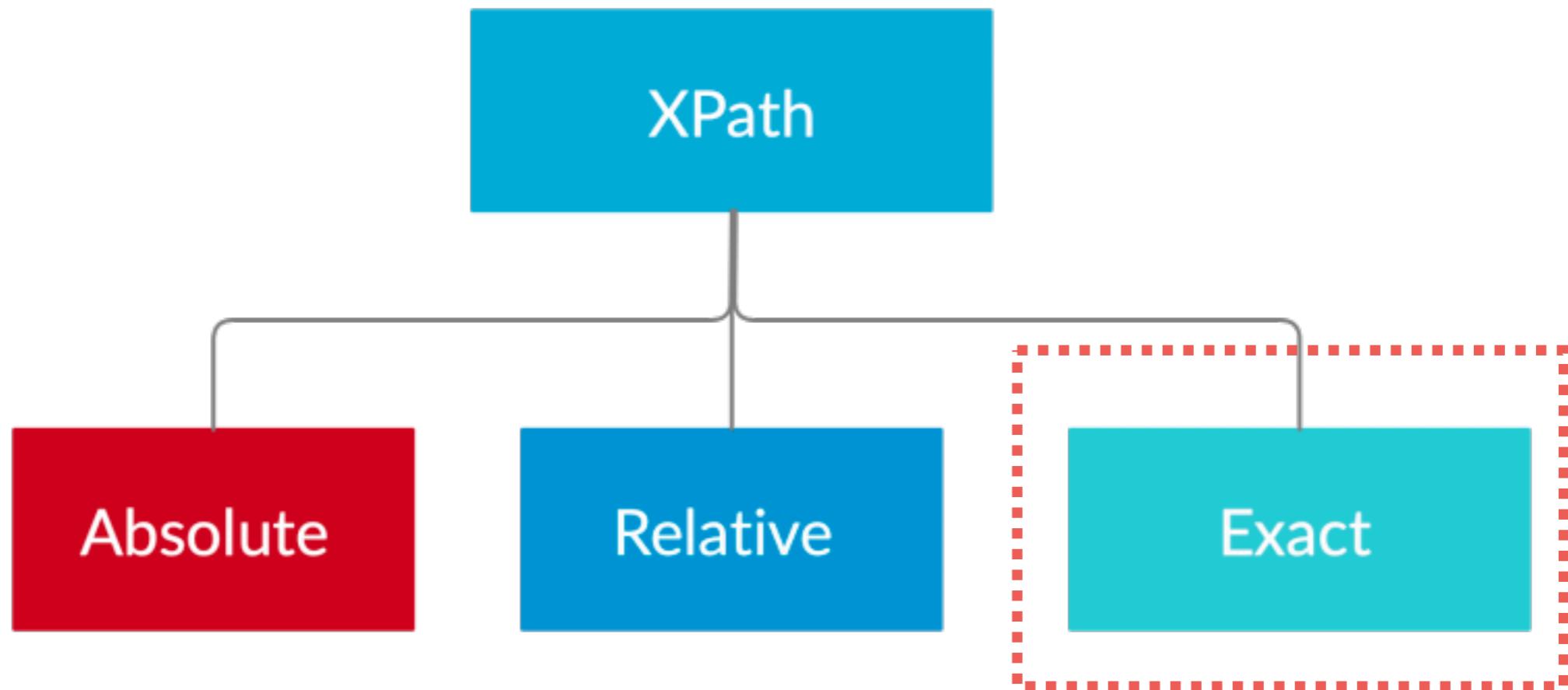
Relative



```
//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

<https://github.com/appium/appium-uiautomator2-driver/blob/master/README.md#mobile-gesture-commands>



Waiting for Elements



Waiting ...

Static waits (sleep) !!

Explicit waits

Wait until the element is found before proceeding
the next step

Wait Until Element Is Visible

Wait Until Page Contains

Wait Until Page Contains Element

Wait Until Page Does Not Contain

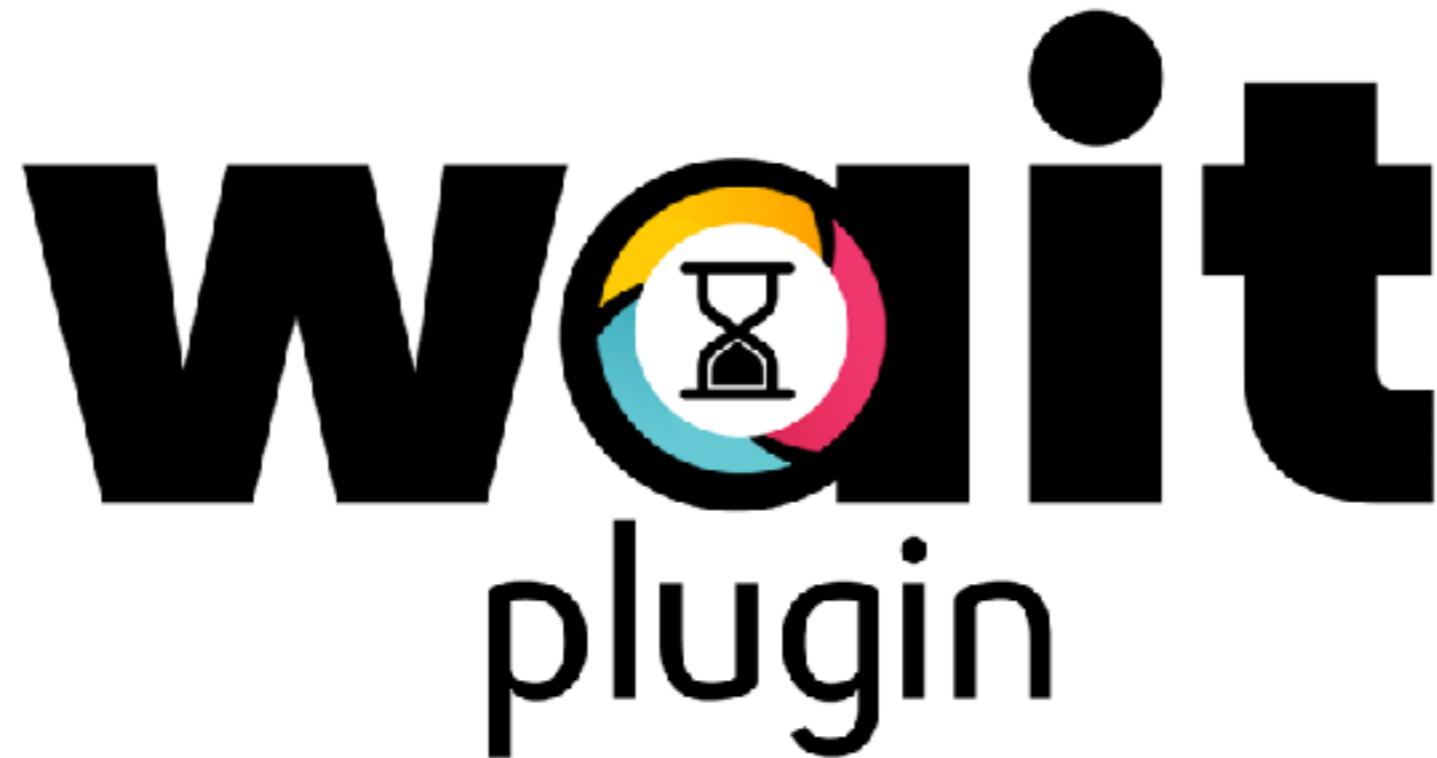
Wait Until Page Does Not Contain Element



Working with Appium's plugin

```
$appium plugin install --source=npm appium-wait-plugin
```

```
$appium --use-plugins=element-wait
```



<https://github.com/AppiumTestDistribution/appium-wait-plugin>



Learn to read Appium logs

```
[HTTP] {}
[AndroidUiautomator2Driver@0c13 (97cf7a81)] Driver proxy active, passing request on via HTTP proxy
[debug] [AndroidUiautomator2Driver@0c13 (97cf7a81)] Matched '/session/97cf7a81-37ce-4ac4-bd05-4d029612237e/source' to command name 'getPageSource'
[debug] [AndroidUiautomator2Driver@0c13 (97cf7a81)] Proxying [GET /session/97cf7a81-37ce-4ac4-bd05-4d029612237e/source] to [GET http://127.0.0.1:8200/session/fe20a07f-45ed-4502-91f5-05df40569d7e/source] with no body
[debug] [AndroidUiautomator2Driver@0c13 (97cf7a81)] Got response with status 200: {"sessionId":"fe20a07f-45ed-4502-91f5-05df40569d7e","value":"<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>\r\n<hierarchy index=\"0\" class=\"hierarchy\" rotation=\"0\" width=\"1080\" height=\"1857\">\r\n  <android.widget.FrameLayout index=\"0\" package=\"com.amazonaws.devicefarm.android.referenceapp\" class=\"android.widget.FrameLayout\" text=\"\" checkable=\"false\" checked=\"false\" clickable=\"false\" enabled=\"true\" focusable=\"false\" focused=\"false\" long-clickable=\"false\" password=\"false\" scrollable=\"false\" selected=\"false\" bounds=\"[0,0][1080,1857]\" displayed=\"true\">\r\n    <android.widget.LinearLayout index=\"0\" package=\"com.amazonaws.devicefarm.android.referenceapp\" class=\"android.widget.LinearLayout\" text=\"\" checkable=\"false\" checked=\"false\" clickable=\"false\" enabled=\"true\" focusable=\"false\" focused=\"false\" long-clickable=\"false\" password=\"false\" scrollable=\"false\" selected=\"false\" bounds=\"[0,0][1080,1857]\" displayed=\"true\">\r\n      <androi...
[AndroidUiautomator2Driver@0c13 (97cf7a81)] Replacing sessionId fe20a07f-45ed-4502-91f5-05df40569d7e with 97cf7a81-37ce-4ac4-bd05-4d029612237e
[HTTP] <-- GET /session/97cf7a81-37ce-4ac4-bd05-4d029612237e/source 200 51 ms - 26619
[HTTP]
[HTTP] --> POST /session/97cf7a81-37ce-4ac4-bd05-4d029612237e/element
```



Write Test cases



Robot Framework



<https://robotframework.org/>





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



Install Appium Library

```
$pip install -U robotframework
```

```
$pip install -U robotframework-appiumlibrary
```

<https://robotframework.org/>



Appium Library keywords

AppiumLibrary

Search X

Keywords (115) +

- Activate Application
- Background App
- Background Application
- Capture Page Screenshot
- Clear Text
- Click A Point
- Click Button
- Click Element
- Click Element At Coordinates
- Click Text
- Close All Applications
- Close Application
- Delete File
- Drag And Drop
- Element Attribute Should Match

Library version: 2.0.0
Library scope: GLOBAL

Introduction

AppiumLibrary is a Mobile App testing library for Robot Framework.

Locating or specifying elements

All keywords in AppiumLibrary that need to find an element on the page take an argument, either a `locator` or a `WebElement`. A `locator` is a string that describes how to locate an element using a syntax specifying different location strategies. A `WebElement` holds a `WebElement` instance, which is a representation of the element.

Using locators

By default, when a locator is provided, it is matched against the key attributes of the particular element. If the `locator` attribute is `id` for all elements and locating elements is easy using just the `id`. For example:

```
Click Element    id=my_element
```

New in AppiumLibrary 1.4, `id` and `xpath` are not required to be specified, however `xpath` should start with `//` to indicate an absolute locator as explained below.

<https://serhatbolsu.github.io/robotframework-appiumlibrary/AppiumLibrary.html>



Locator Elements

Strategy	Example	Description
identifier	Click Element / identifier=my_element	Matches by @id attribute
id	Click Element / id=my_element	Matches by @resource-id attribute
accessibility_id	Click Element / accessibility_id=button3	Accessibility options utilize.
xpath	Click Element / xpath=//UITableView/UIATableCell/UIAButton	Matches with arbitrary XPath
class	Click Element / class=UIAPickerWheel	Matches by class
android	Click Element / android=UiSelector().description('Apps')	Matches by Android UI Automator
ios	Click Element / ios=.buttons().withName('Apps')	Matches by iOS UI Automation
nsp	Click Element / nsp=name=="login"	Matches by iOSNsPredicate
chain	Click Element / chain=XCUIElementTypeWindow[1]/*	Matches by iOS Class Chain
css	Click Element / css=.green_button	Matches by css in webview
name	Click Element / name=my_element	Matches by @name attribute

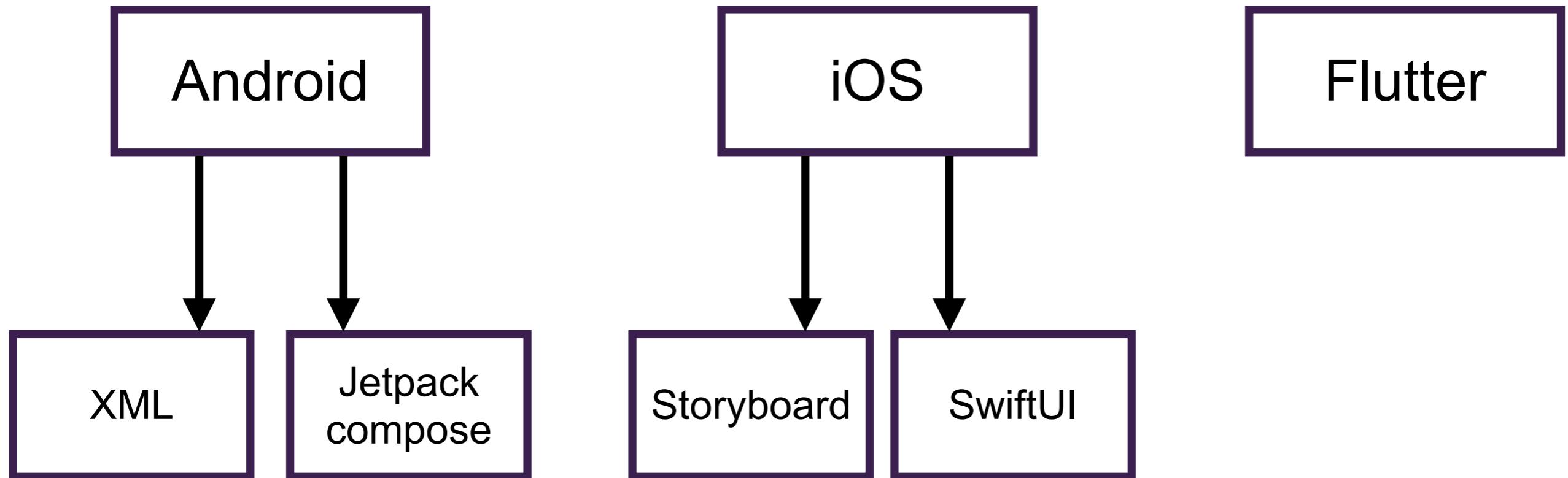
<https://serhatbolsu.github.io/robotframework-appiumlibrary/AppiumLibrary.html>



Write your tests



Target App with UI Test



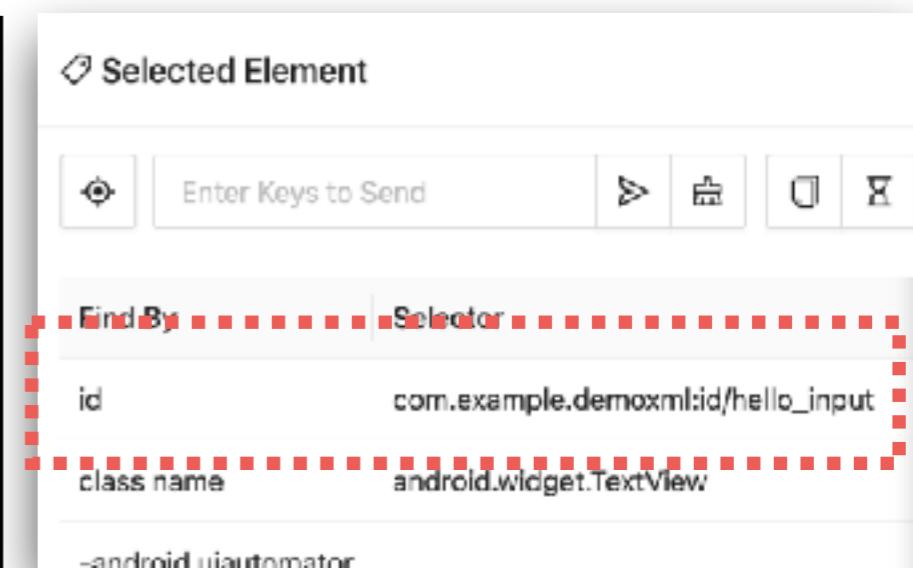
Android + Jetpack Compose

MainActivity.kt

```
@OptIn(ExperimentalComposeUiApi::class)
fun Modifier.setTagAndId(tag: String): Modifier {
    return this
        .semantics { this.testTagsAsResourceId = true }
        .testTag(tag)
}

@Composable
fun Greeting(name: String, modifier: Modifier = Modifier) {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.Center
    ) {
        Text(
            text = "XXXXXXHello $name!",
            modifier =
                Modifier.setTagAndId("com.example.demoxml:id/hello_input")
        )
    }
}
```

Appium Inspector

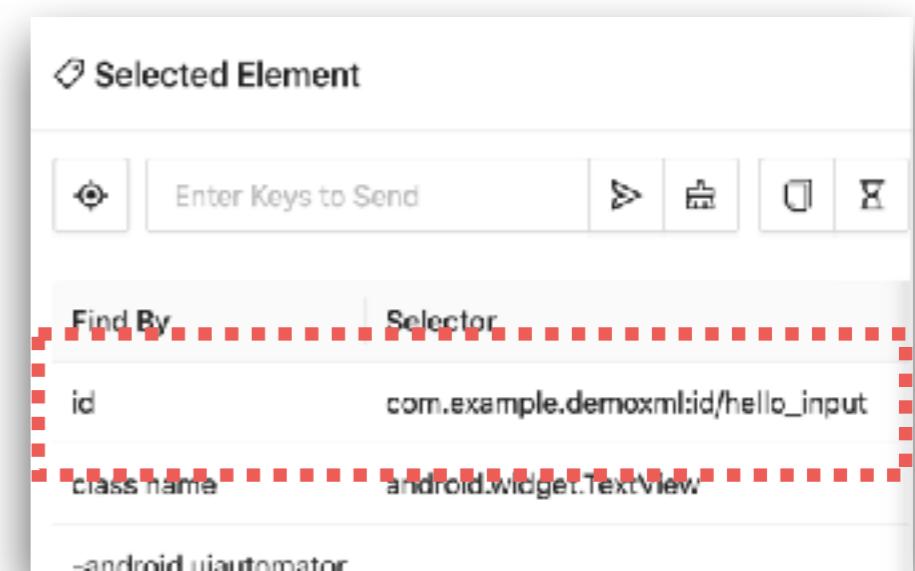


Android + XML Layout

activity_main.xml

```
<TextView  
    android:id="@+id/hello_input"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Hello World from xml"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />
```

Appium Inspector



appPackage=com.example.demoxml



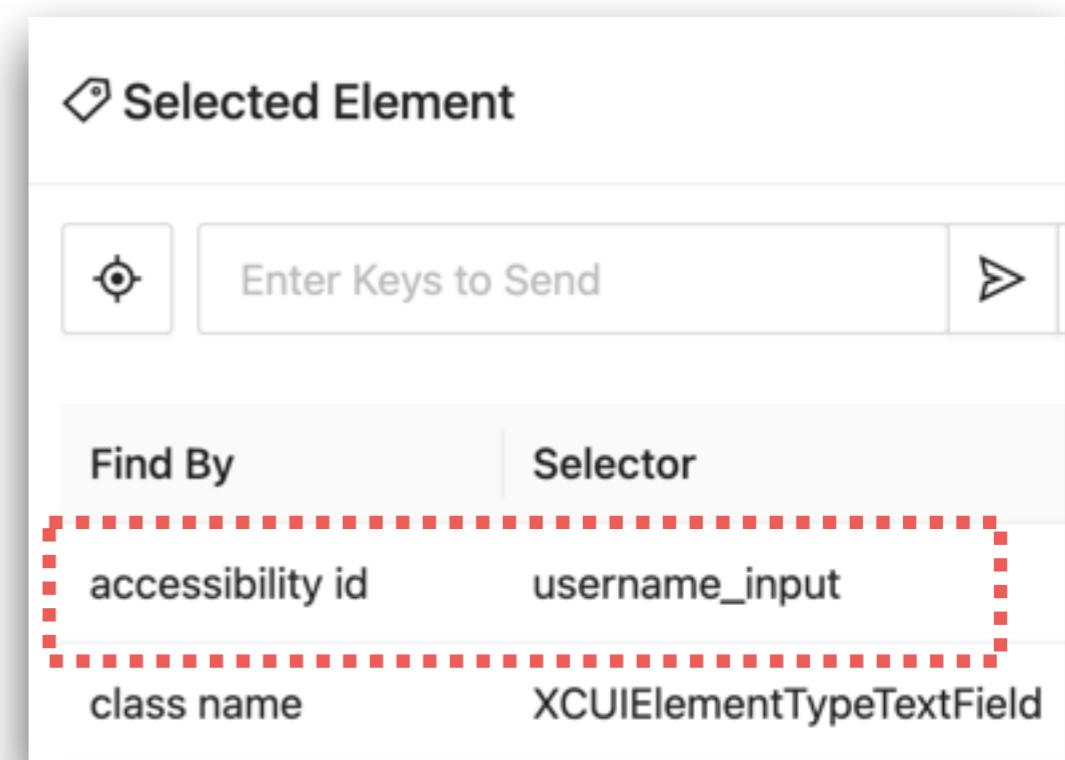
iOS + SwiftUI

LoginView.swift

```
TextField("Username", text: $username)
    .textFieldStyle(RoundedBorderTextFieldStyle())
    .padding()
    .accessibilityIdentifier("username_input")

SecureField("Password", text: $password)
    .textFieldStyle(RoundedBorderTextFieldStyle())
    .padding()
    .accessibilityIdentifier("password_input")
```

Appium Inspector



Flaky test in UI automation

<https://www.browserstack.com/test-observability/features/test-reporting/what-is-flaky-test>



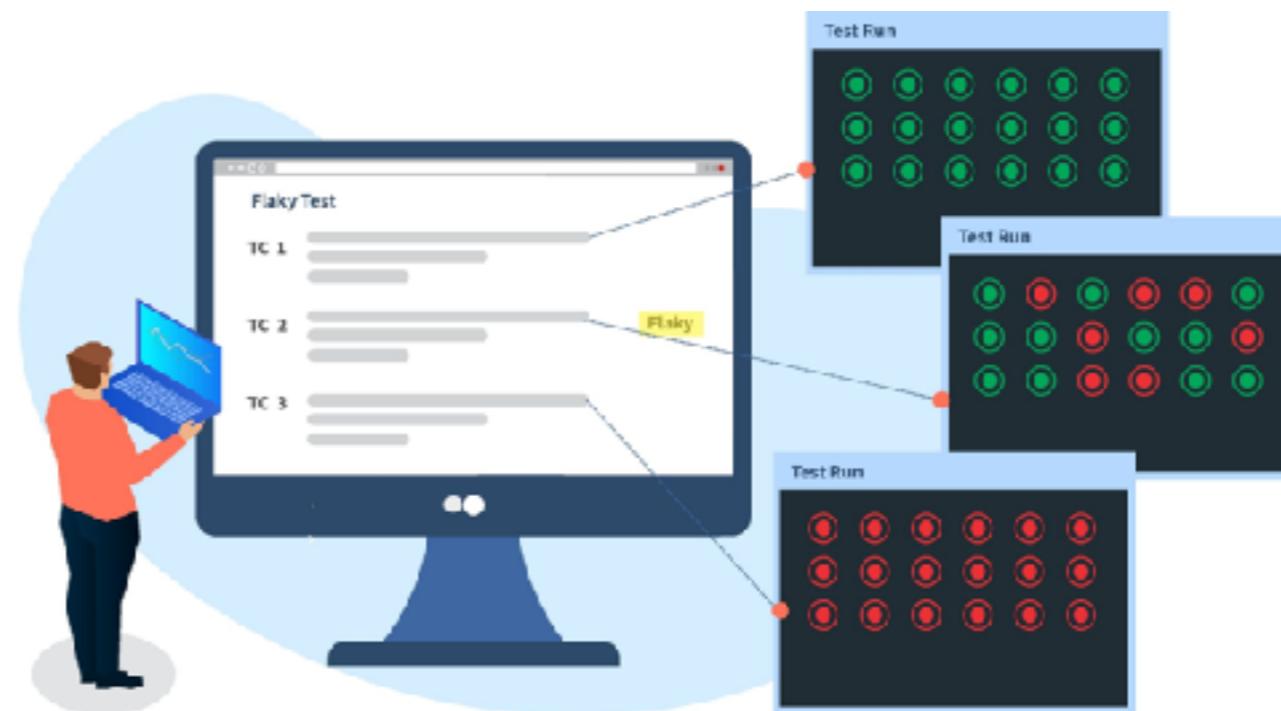
Flaky test

Inconsistency test results

Non-repeatable

Pass or failure unpredictable

System not change !!



Causes of Flaky test

Concurrency
issues

Insufficient
assertions

Flaw logic

Unstable
environment

External
dependencies

Non-deterministic
behavior



How to detect Flaky test ?

Re-run only failed tests

Run tests in parallel way

Run test in different environments

Analyze test results and logs

Use tools and frameworks

Customize
waiting

Parallel testing



How to fix Flaky test ?

Isolate test

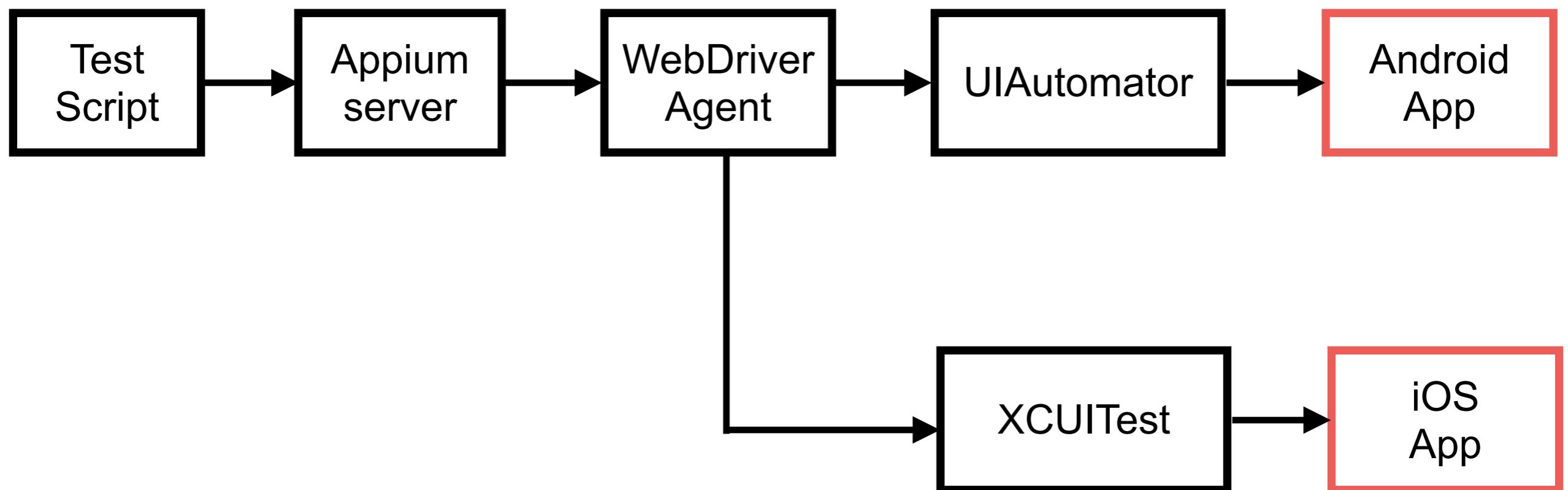
Eliminate randomness

Increase robustness of test case

Simplify the logic of test scripts



Flaky test with Appium !!



Make it Right

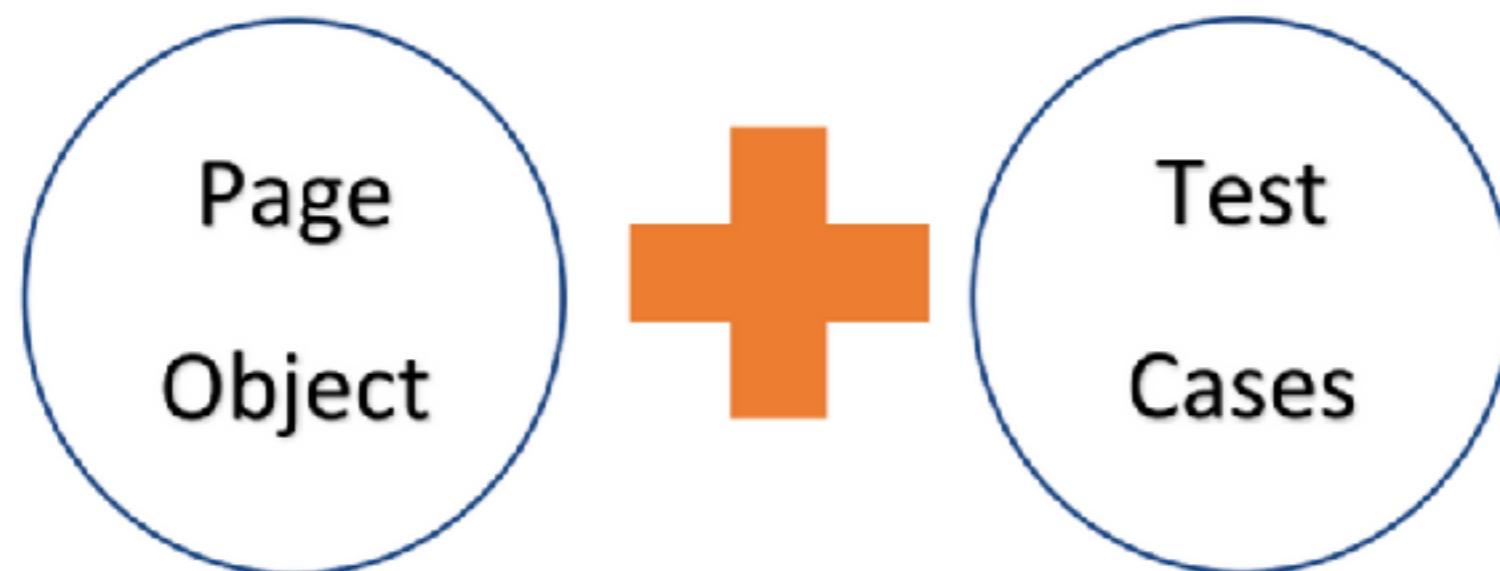


Page Object Pattern

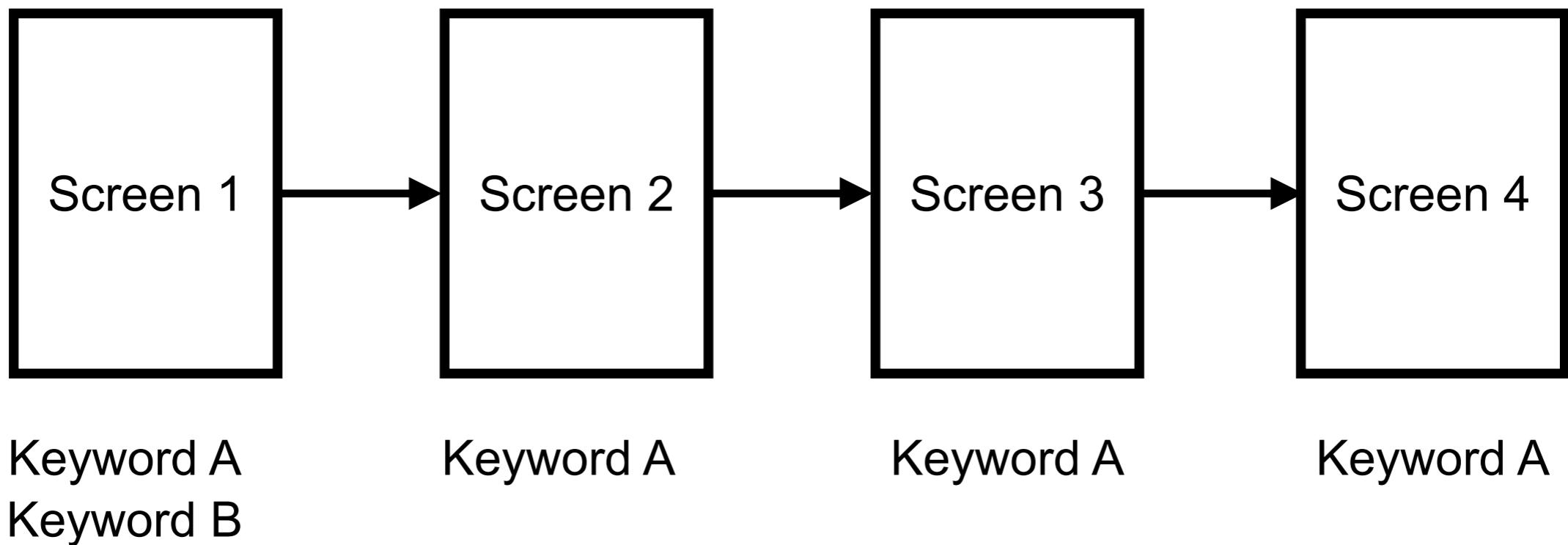


Page Object Pattern

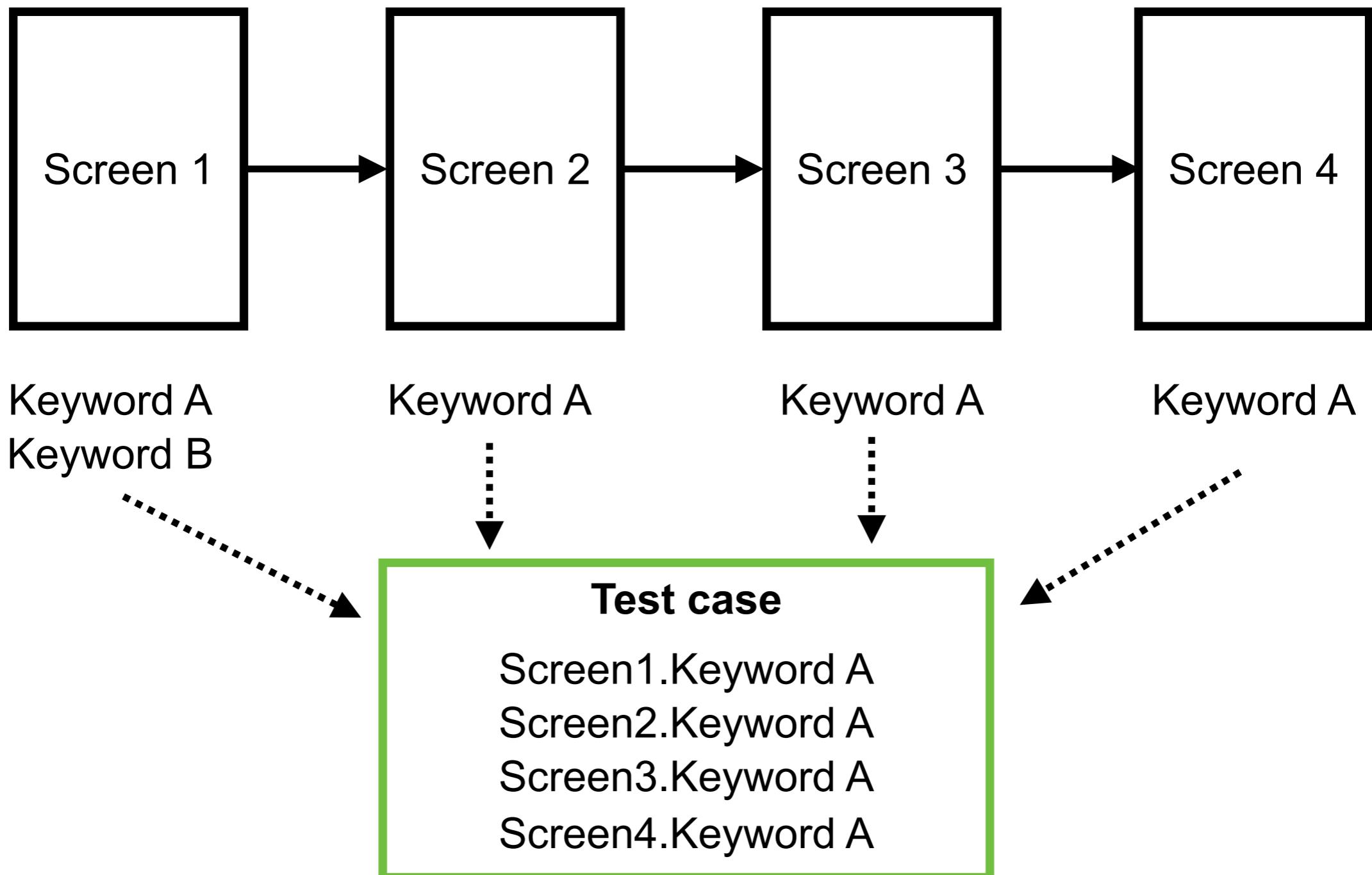
Maintainable test cases
Reduce code duplication
Working as a team



Use Page Object Pattern



Use Page Object Pattern

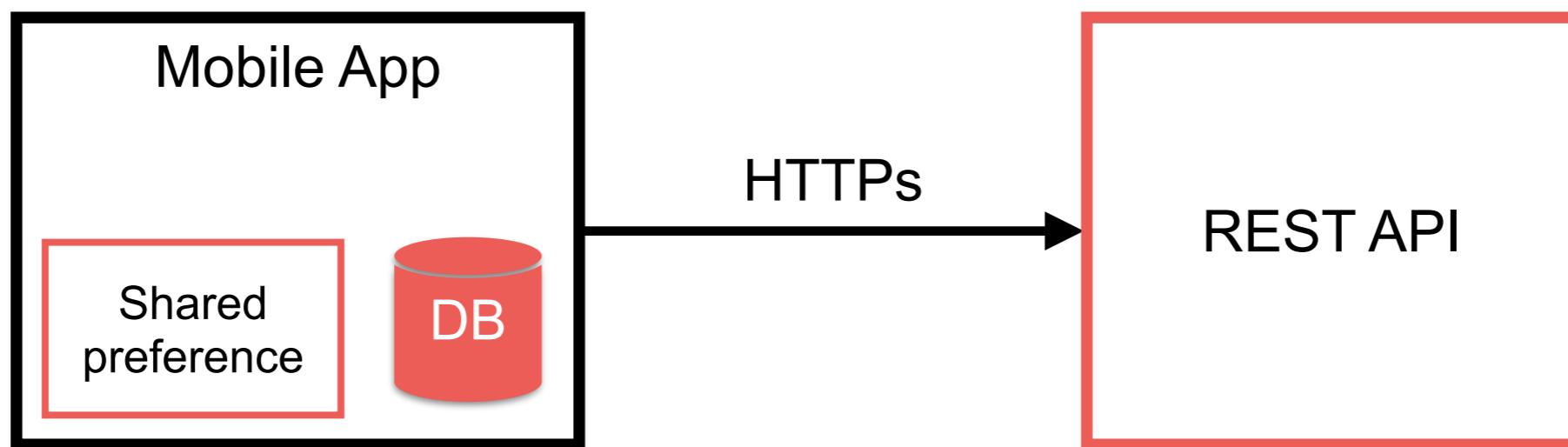




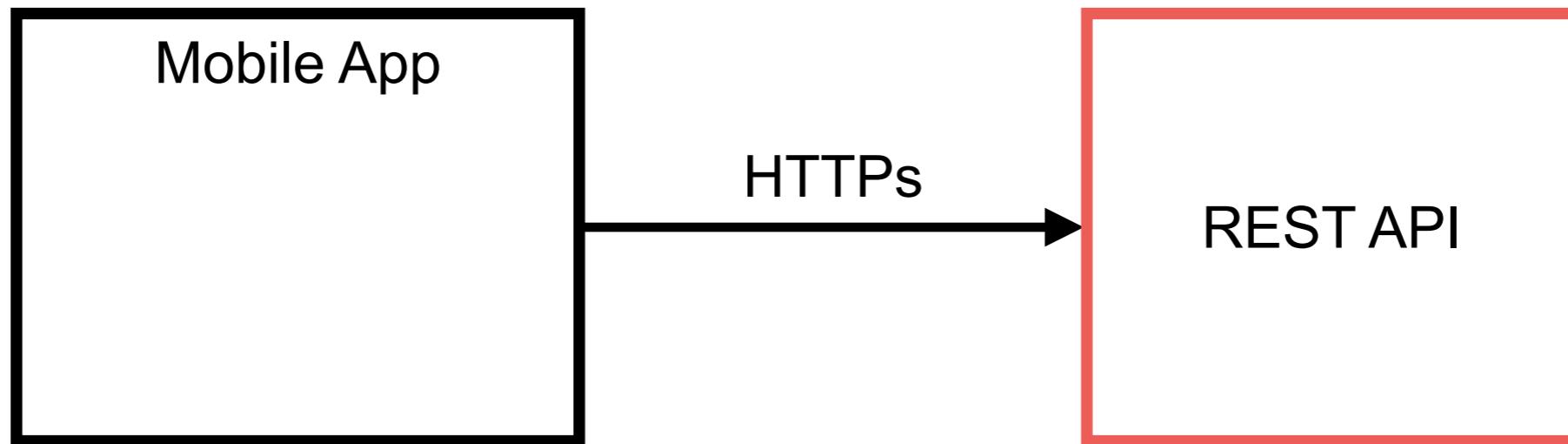
Manage dependencies



Manage dependencies



Manage REST API ?

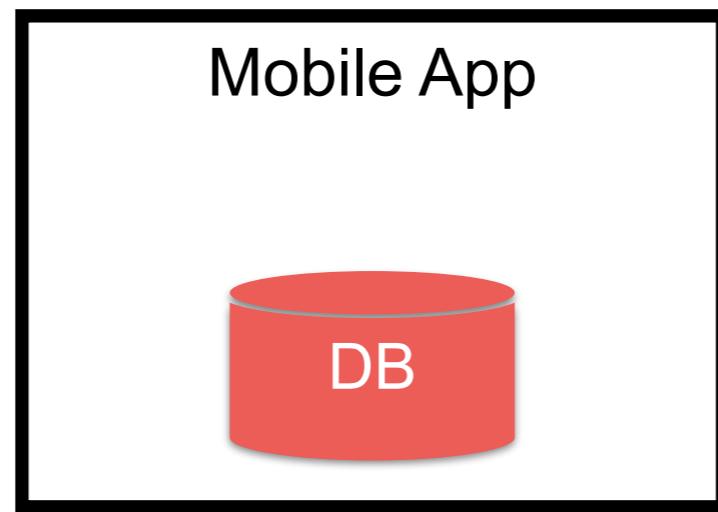


1. Network status ?
2. Mock API server (success/failure) ?
3. Intercept request ?

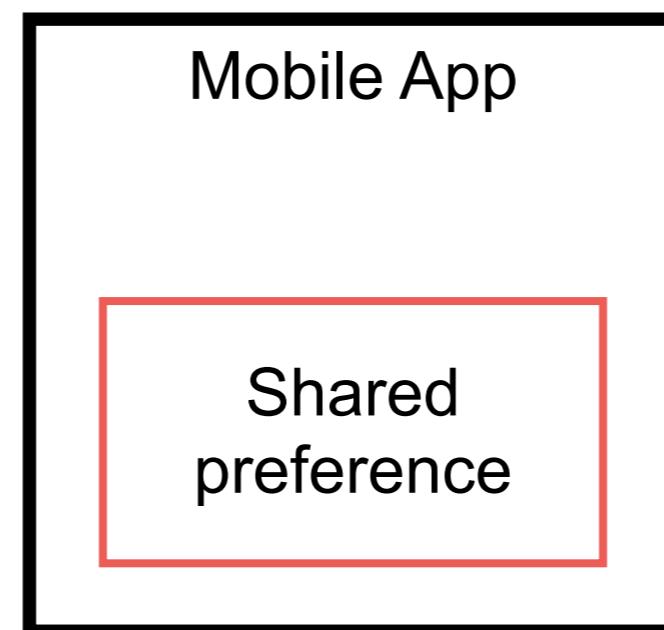




Manage Database ?



Manage SharedPreference ?

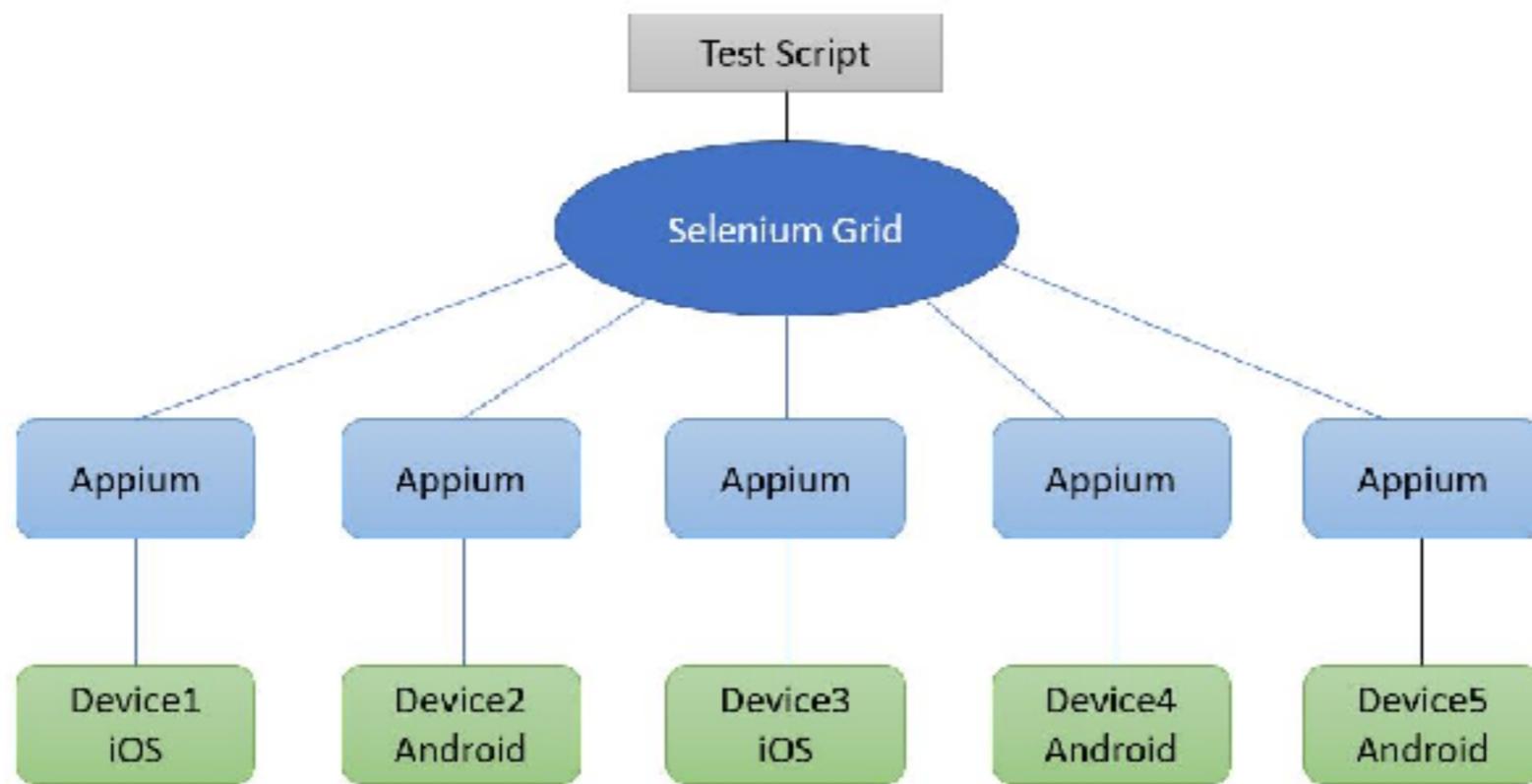


Make it Fast



Appium Grid

Proxy server that allow to run parallel tests on multiple devices simultaneously



<https://appium.io/docs/en/2.1/guides/grid/>
<https://github.com/appium/appium-docker-android>



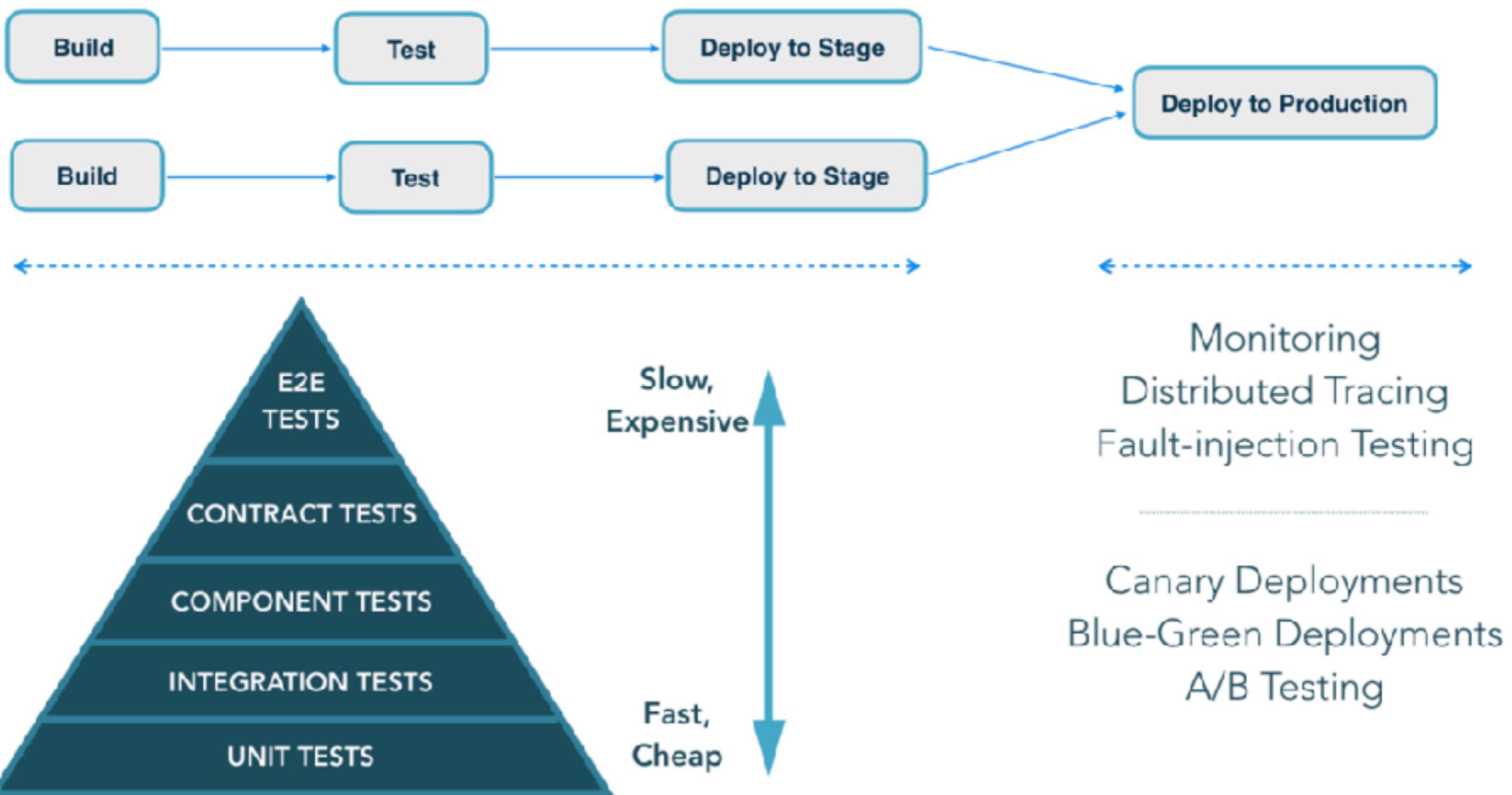
Continuous Integration



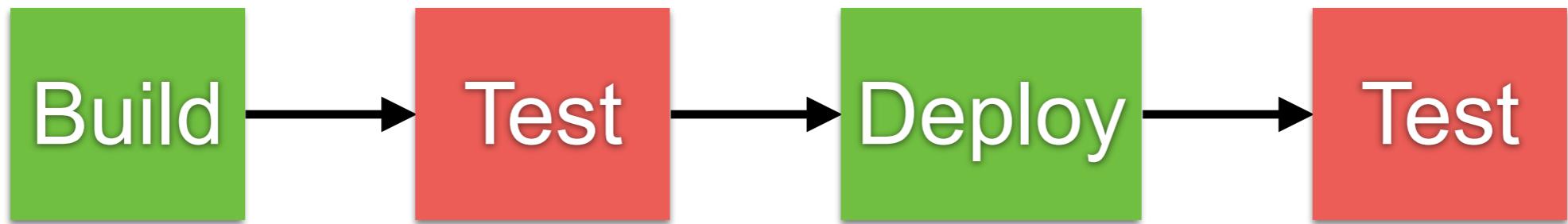
Continuous integration



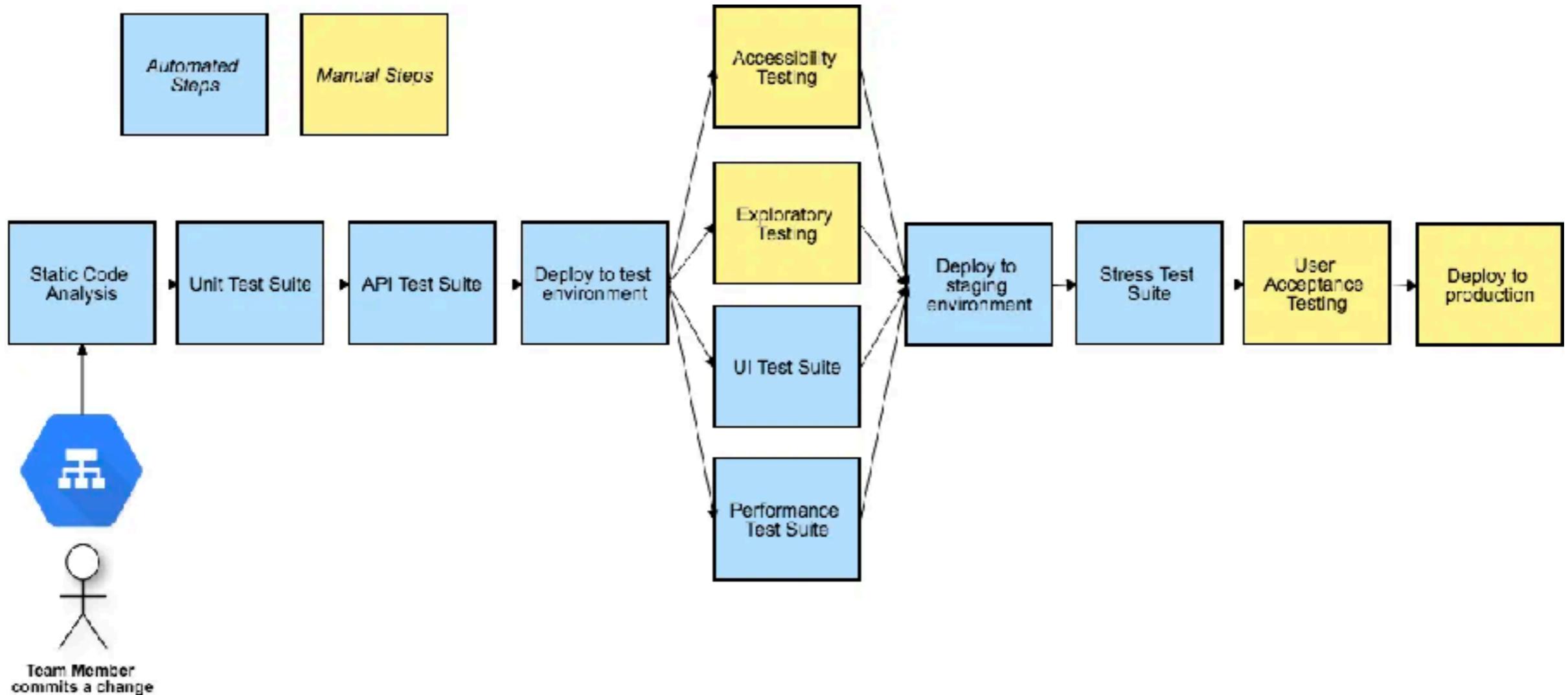
Test strategy



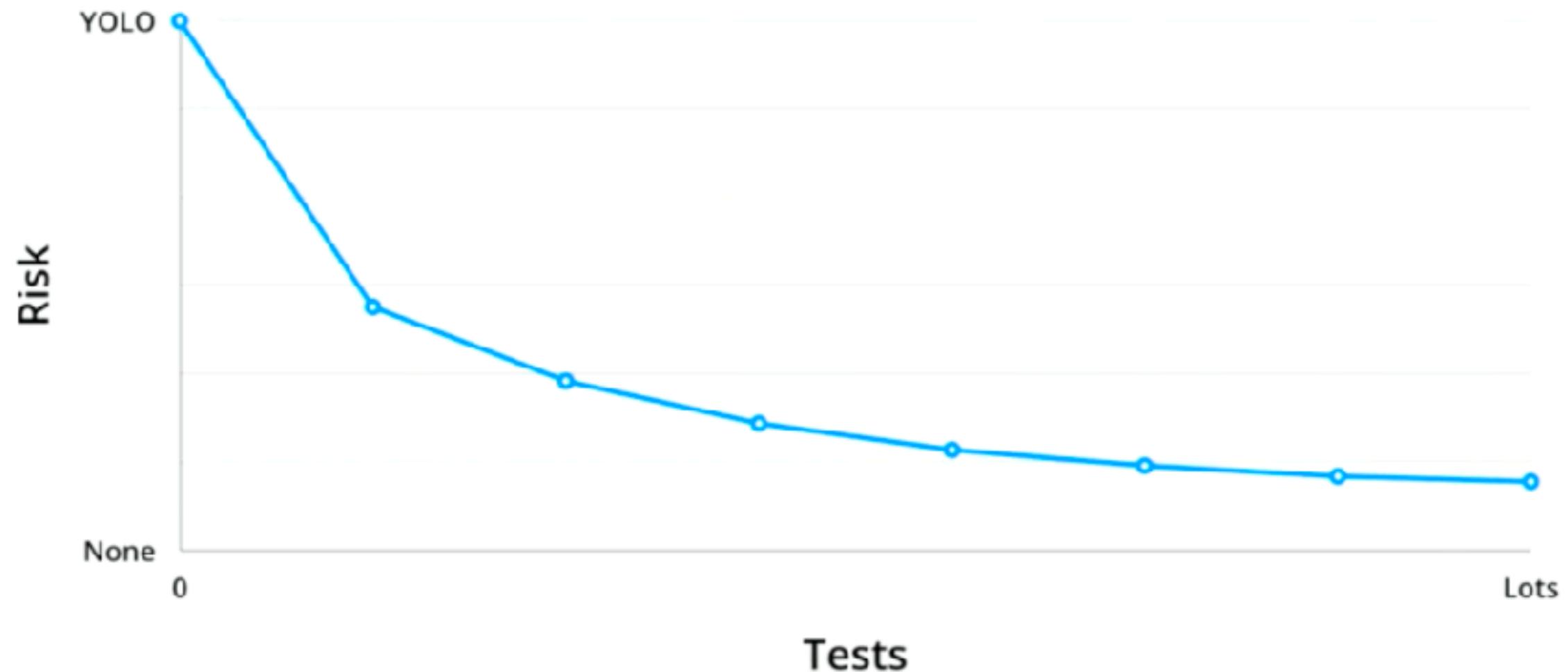
Design your pipeline



Design your pipeline/process



Reduce risk with tests



Start with your journey

