



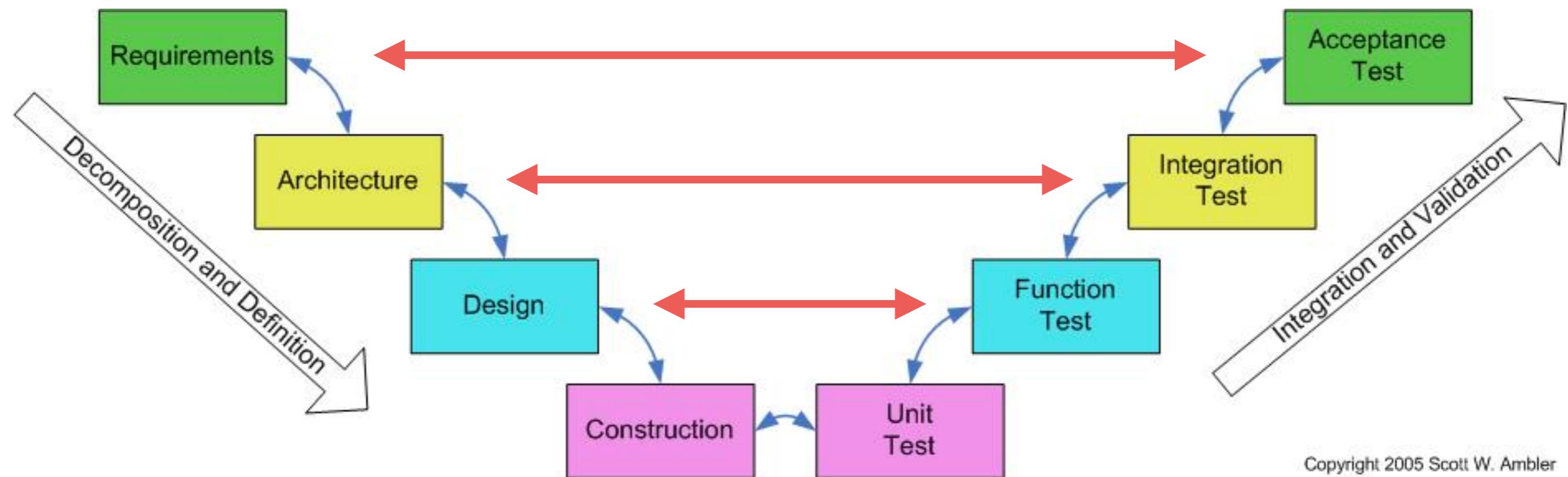
# Automated testing with Katalon and Robot framework



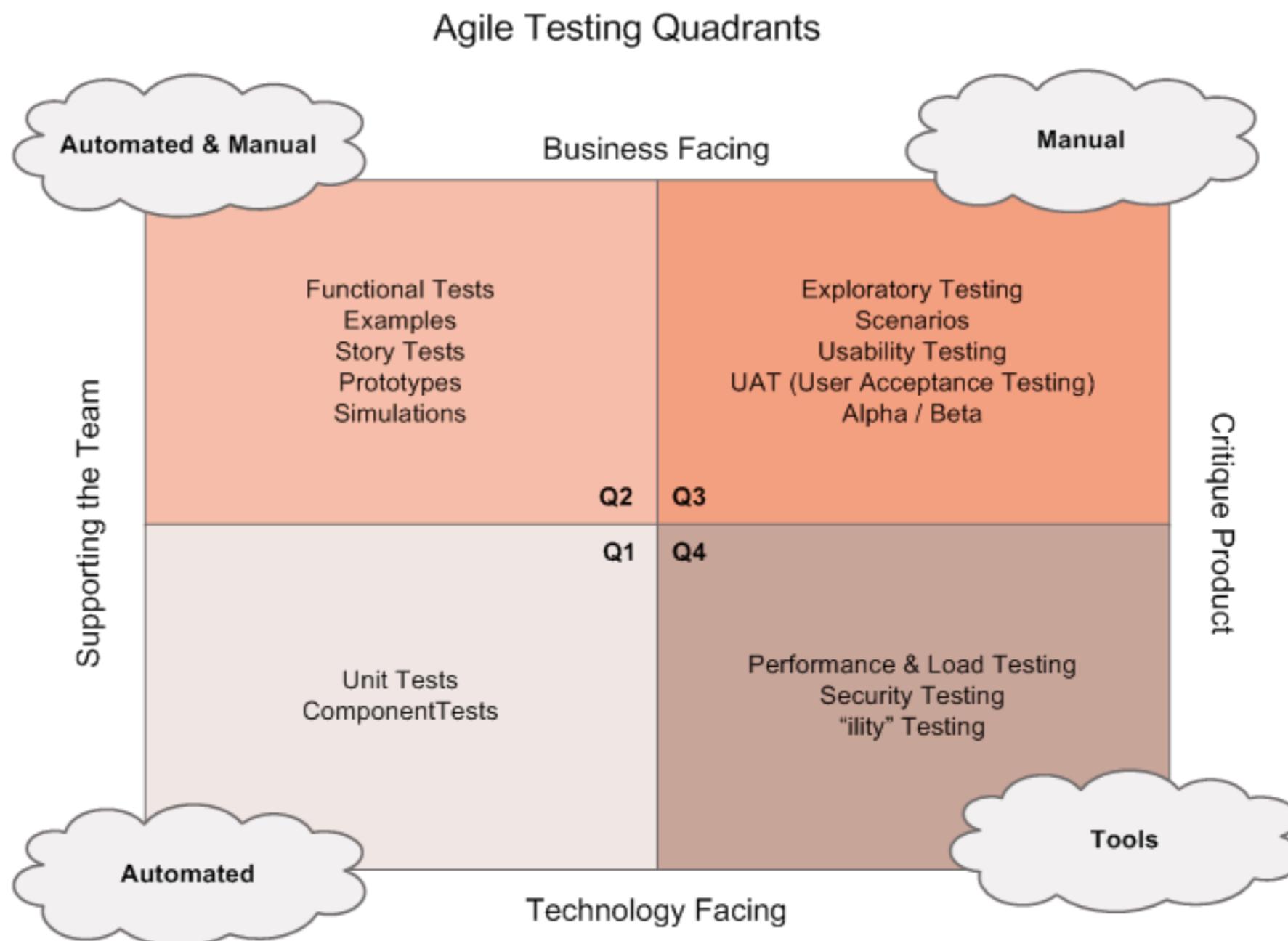
# Automation Testing



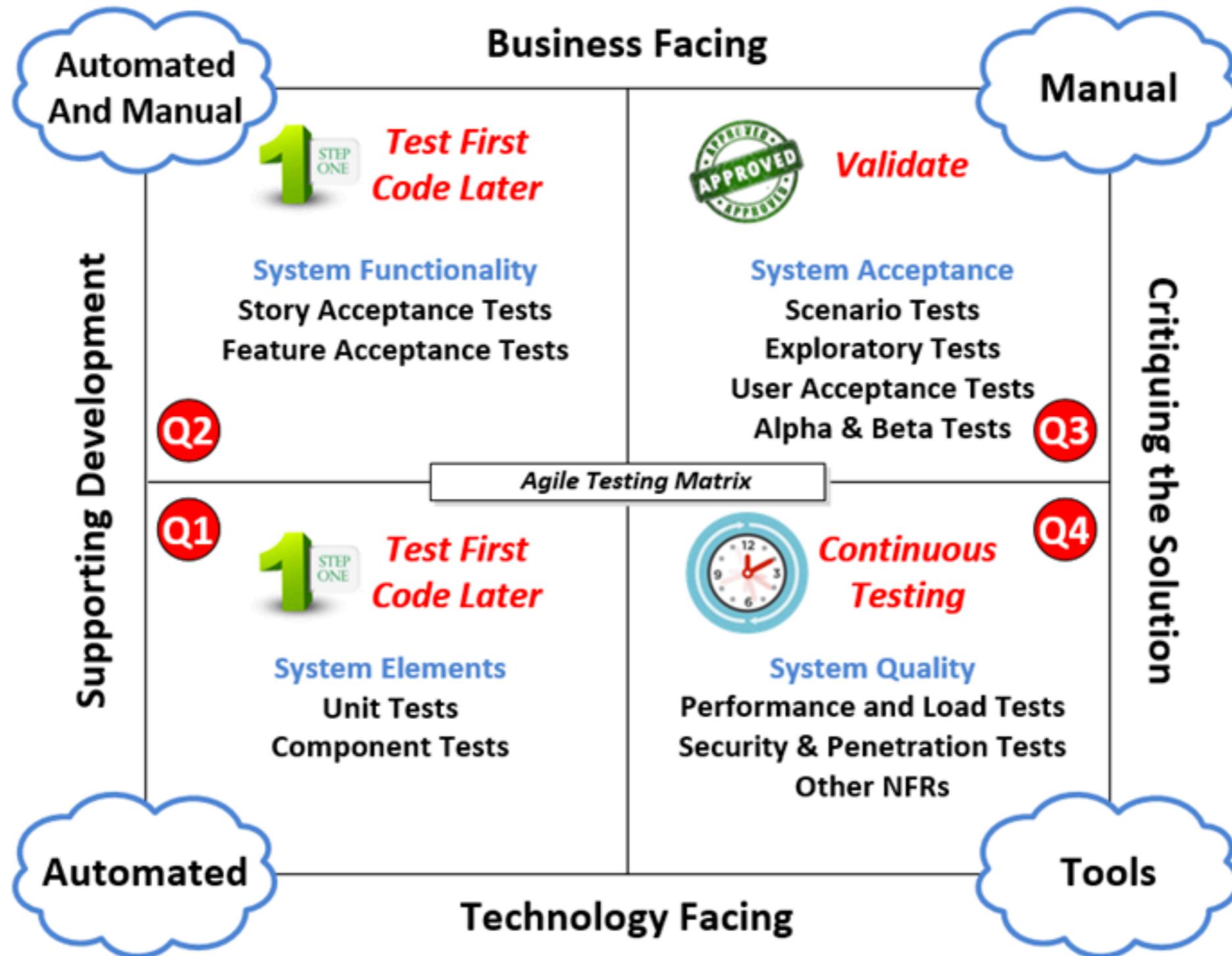
# V Model



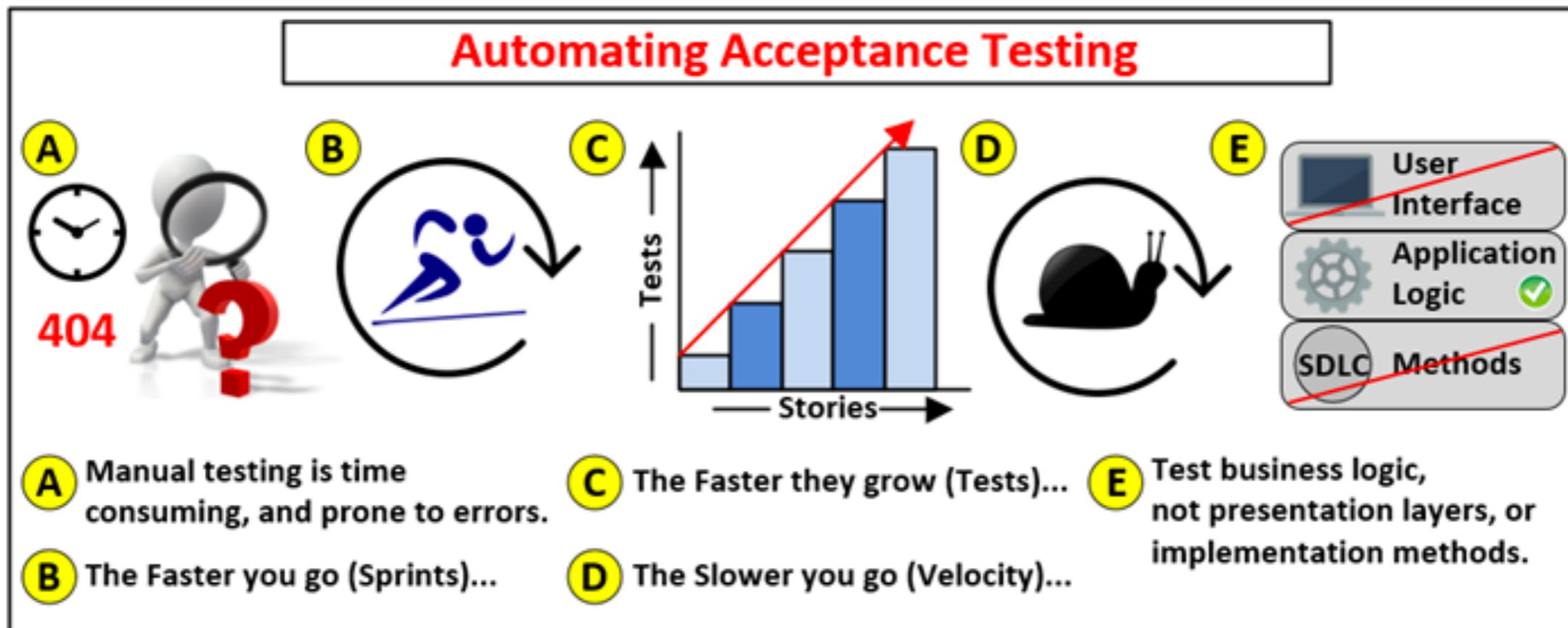
# Agile Testing Quadrant



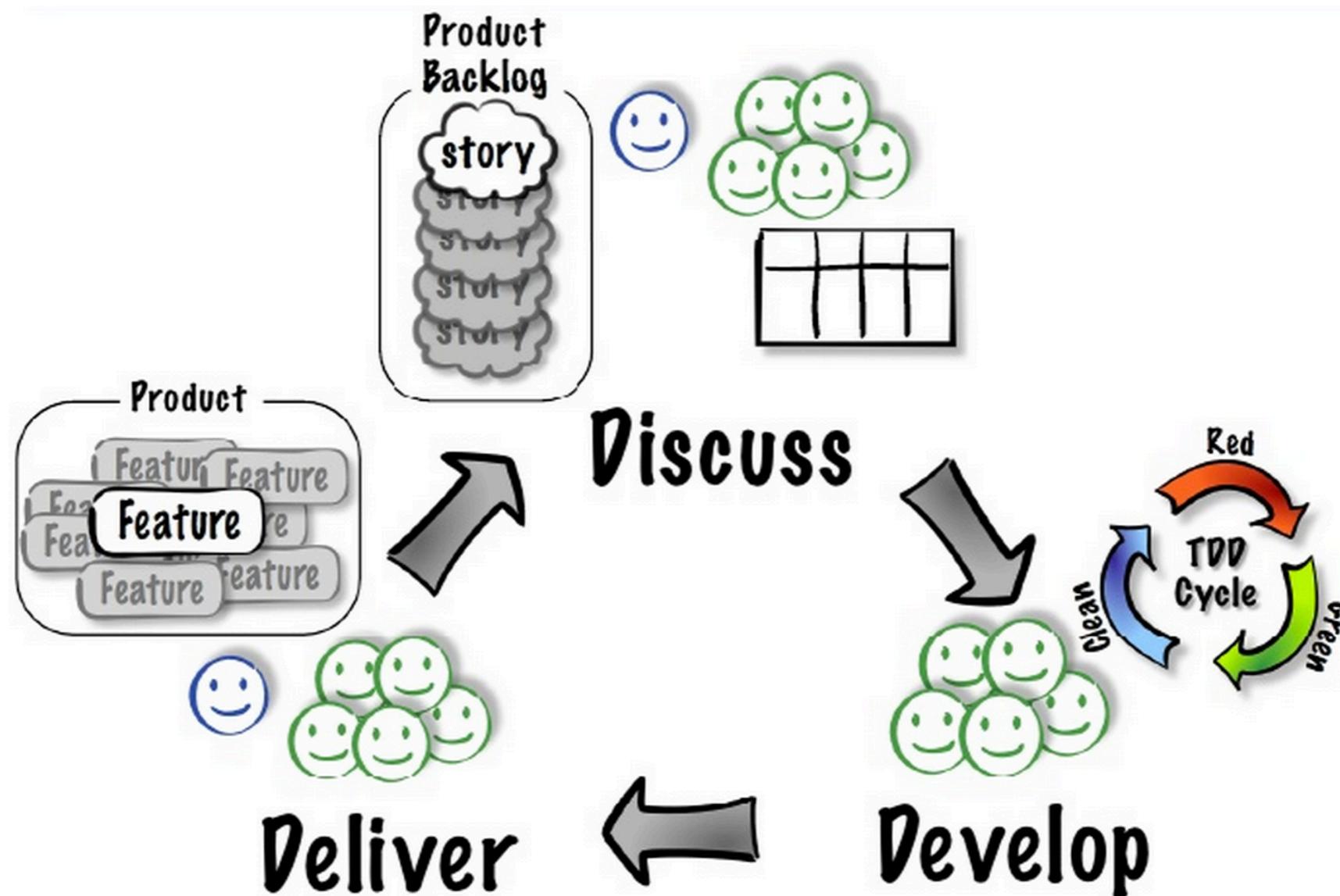
# Agile Testing Quadrant



# Test automation is essential



# Acceptance Test-Driven Development



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



# ATDD

Common language  
Common and share understanding  
Executable requirements or examples  
Living document



# **Acceptance Tests**

=

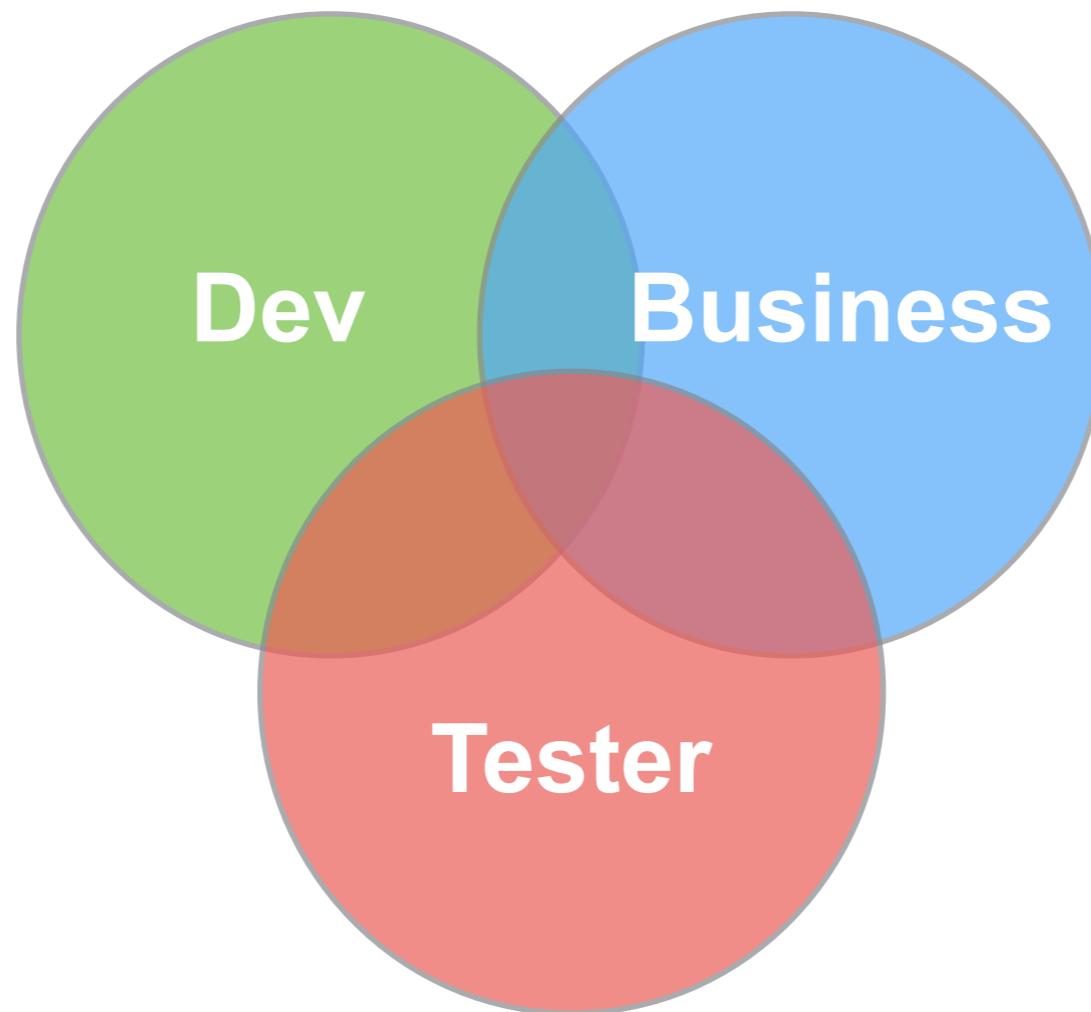
## **Business Criteria**

+

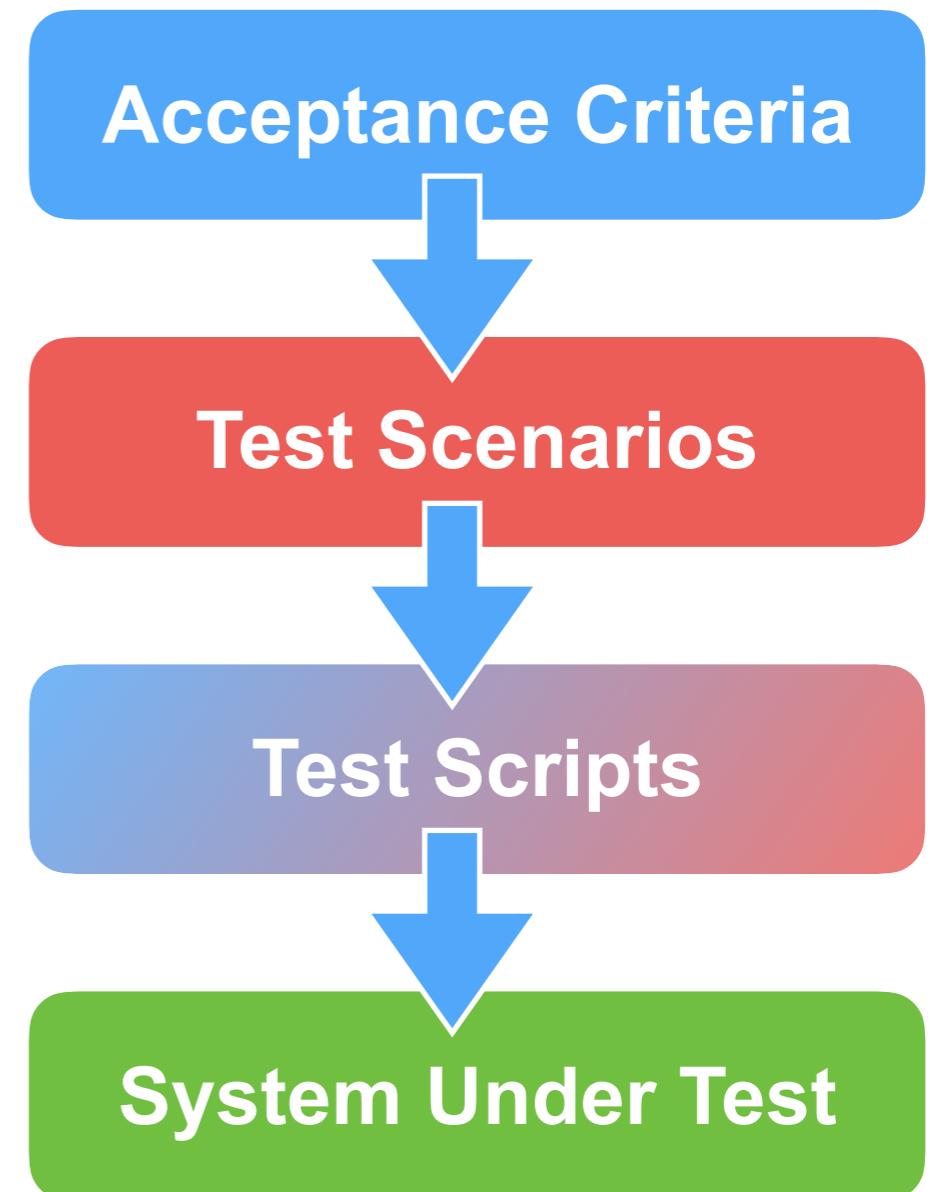
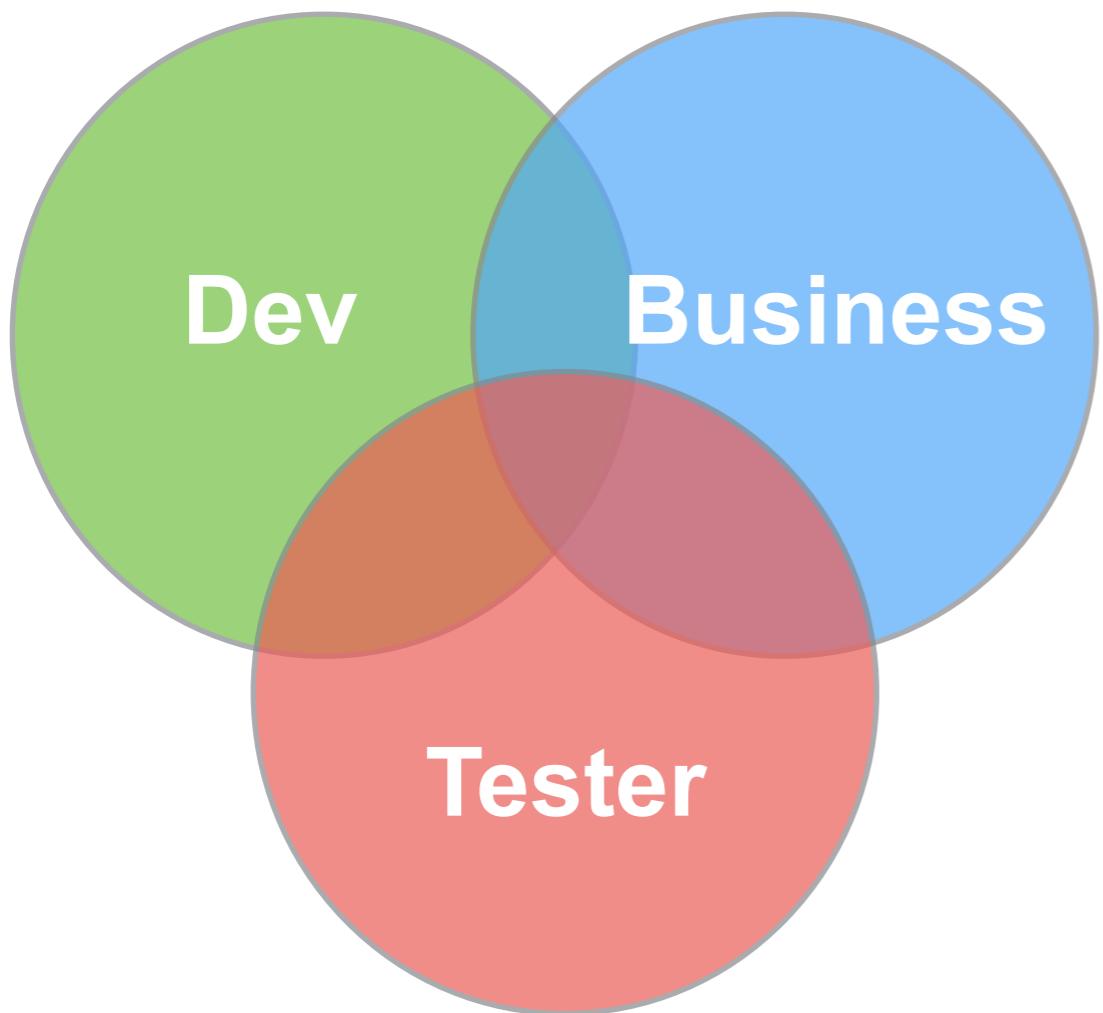
## **Examples (data)**



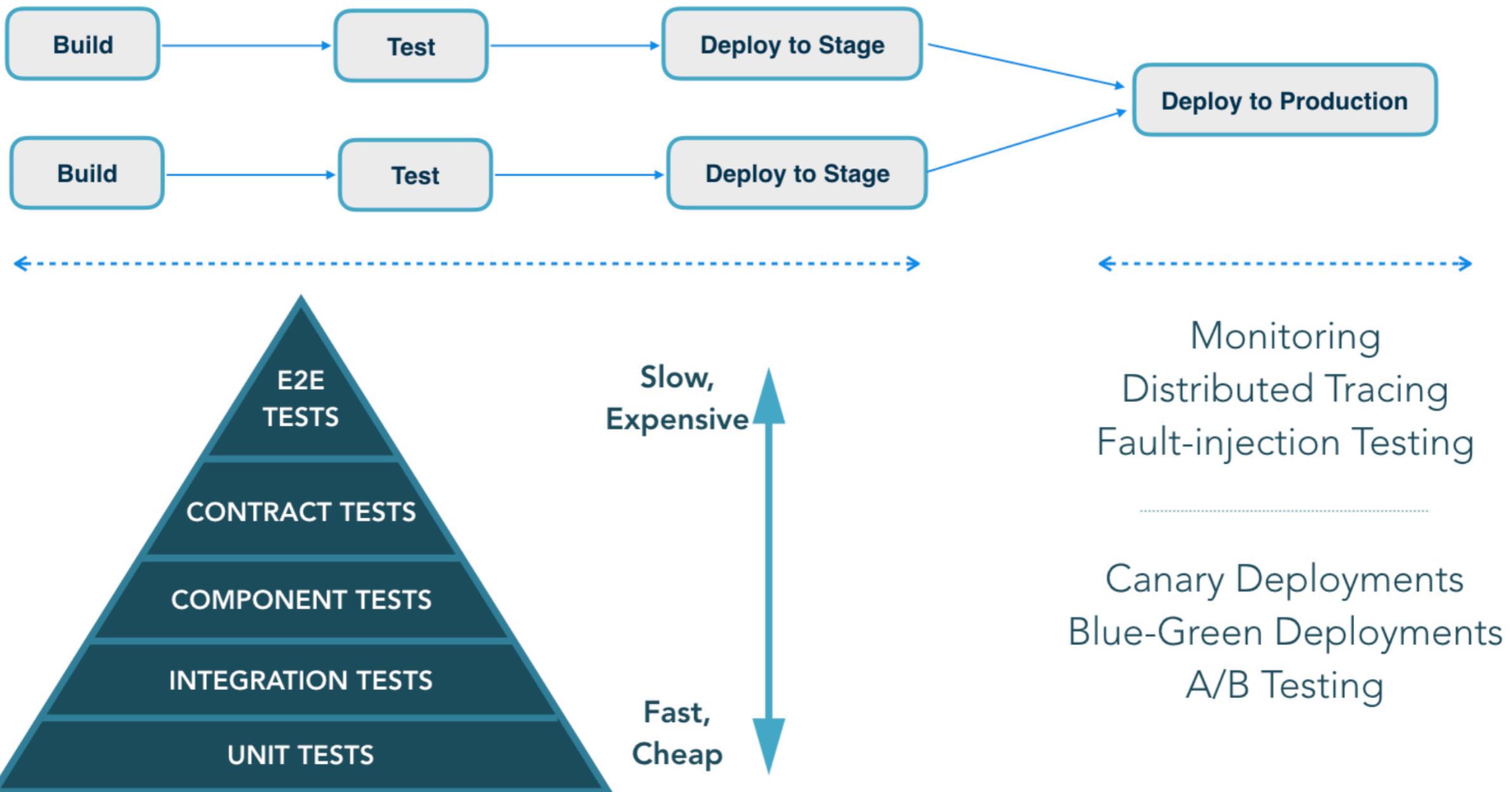
# Acceptance testing



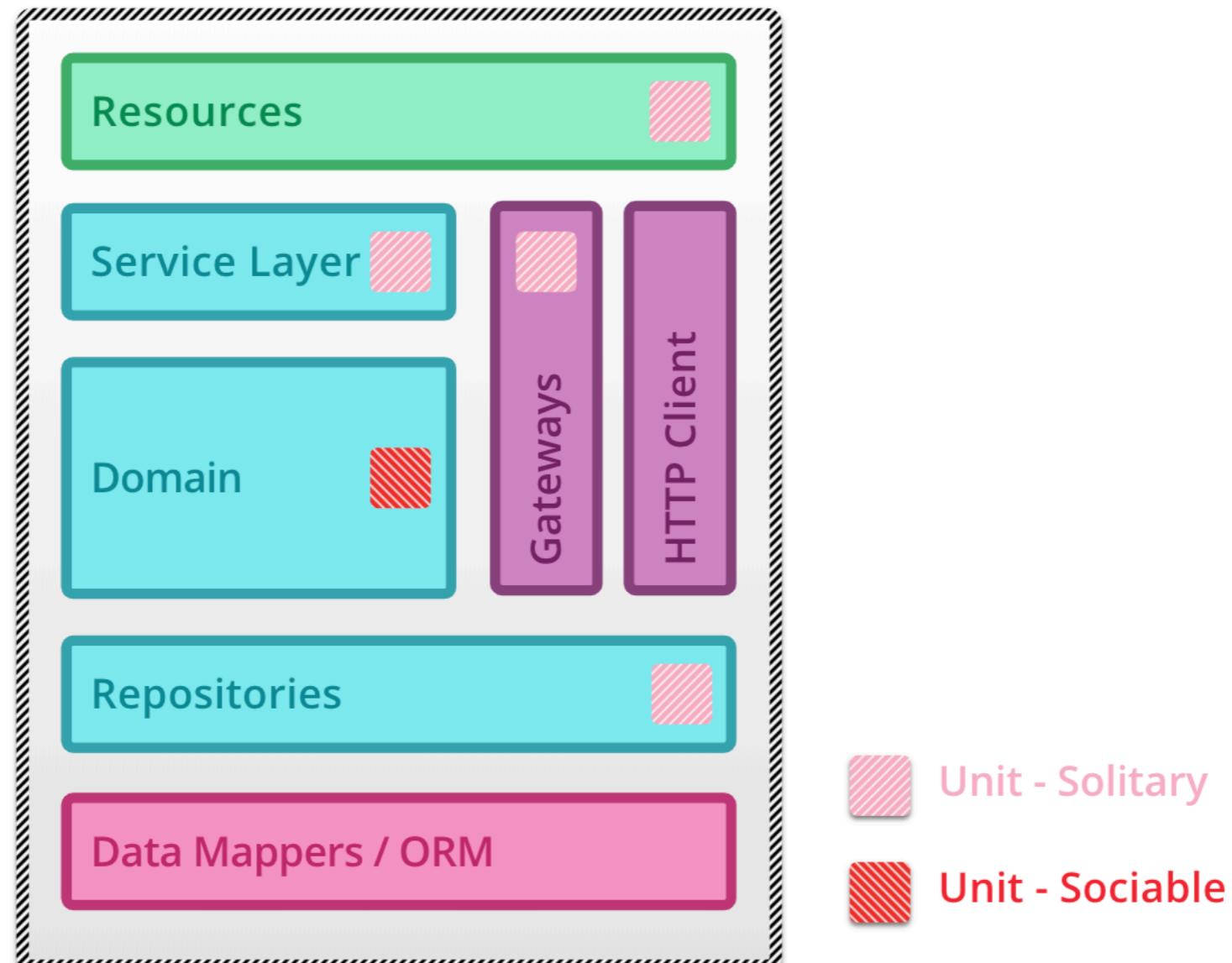
# Acceptance testing



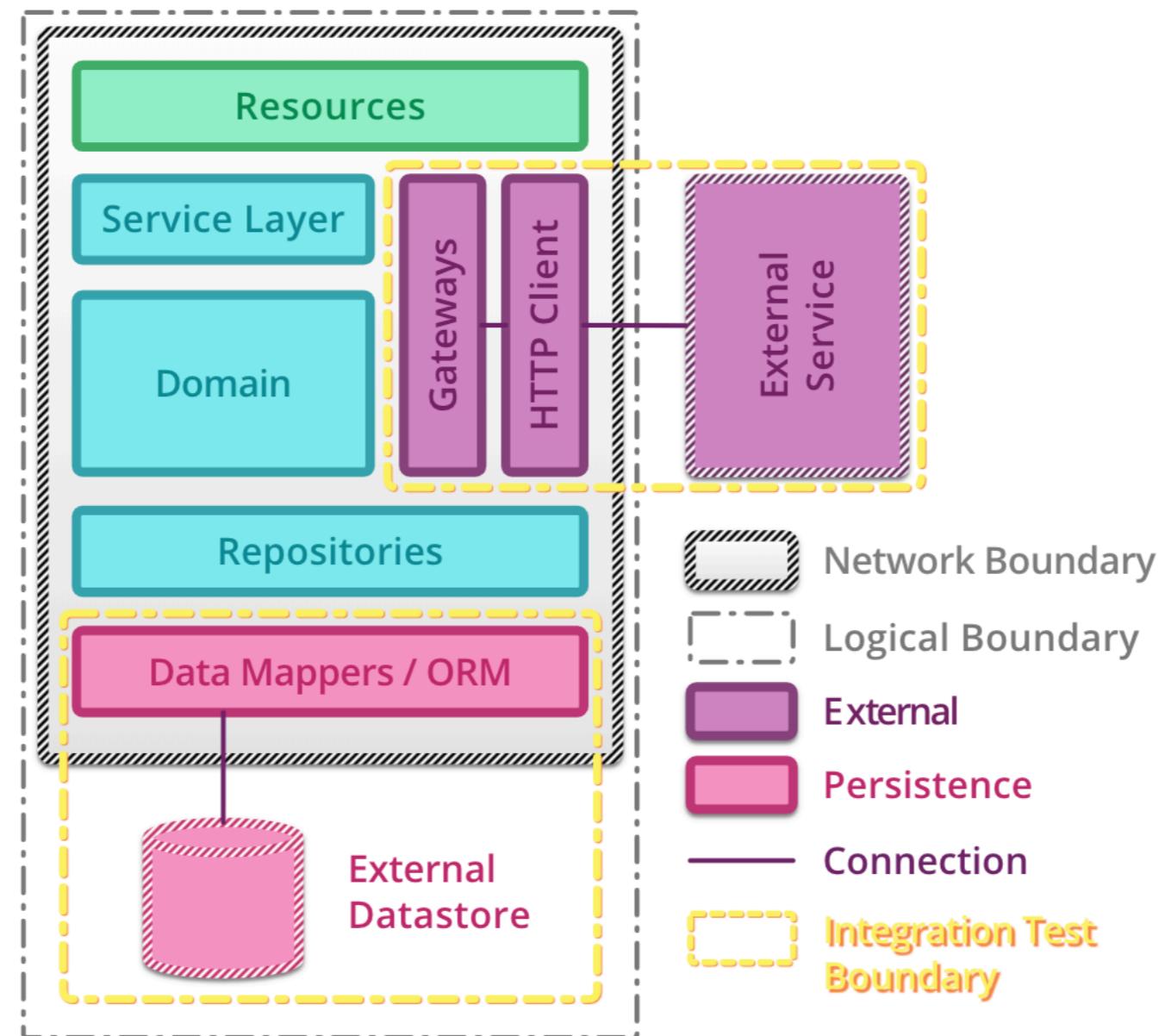
# Test strategy



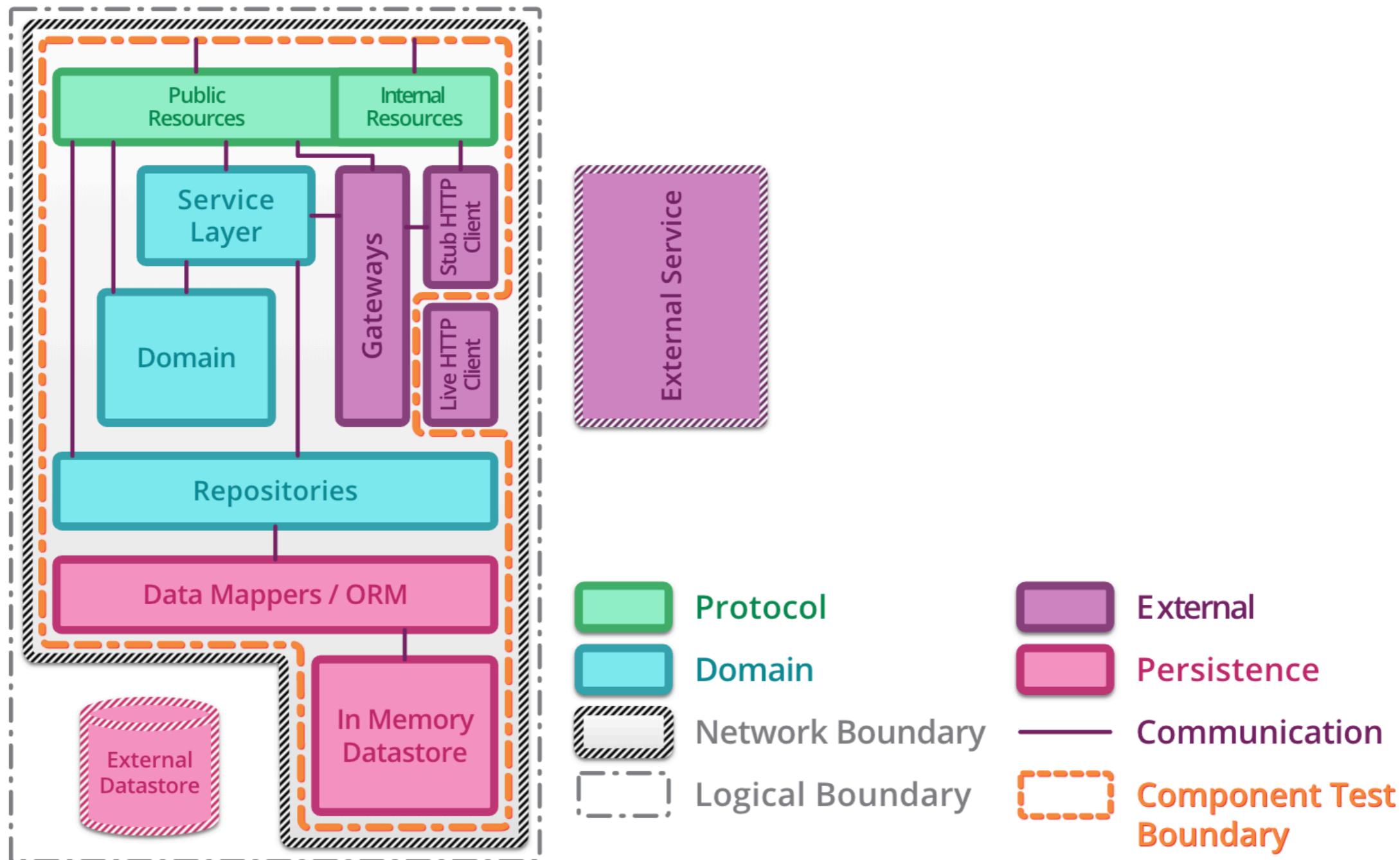
# Unit testing



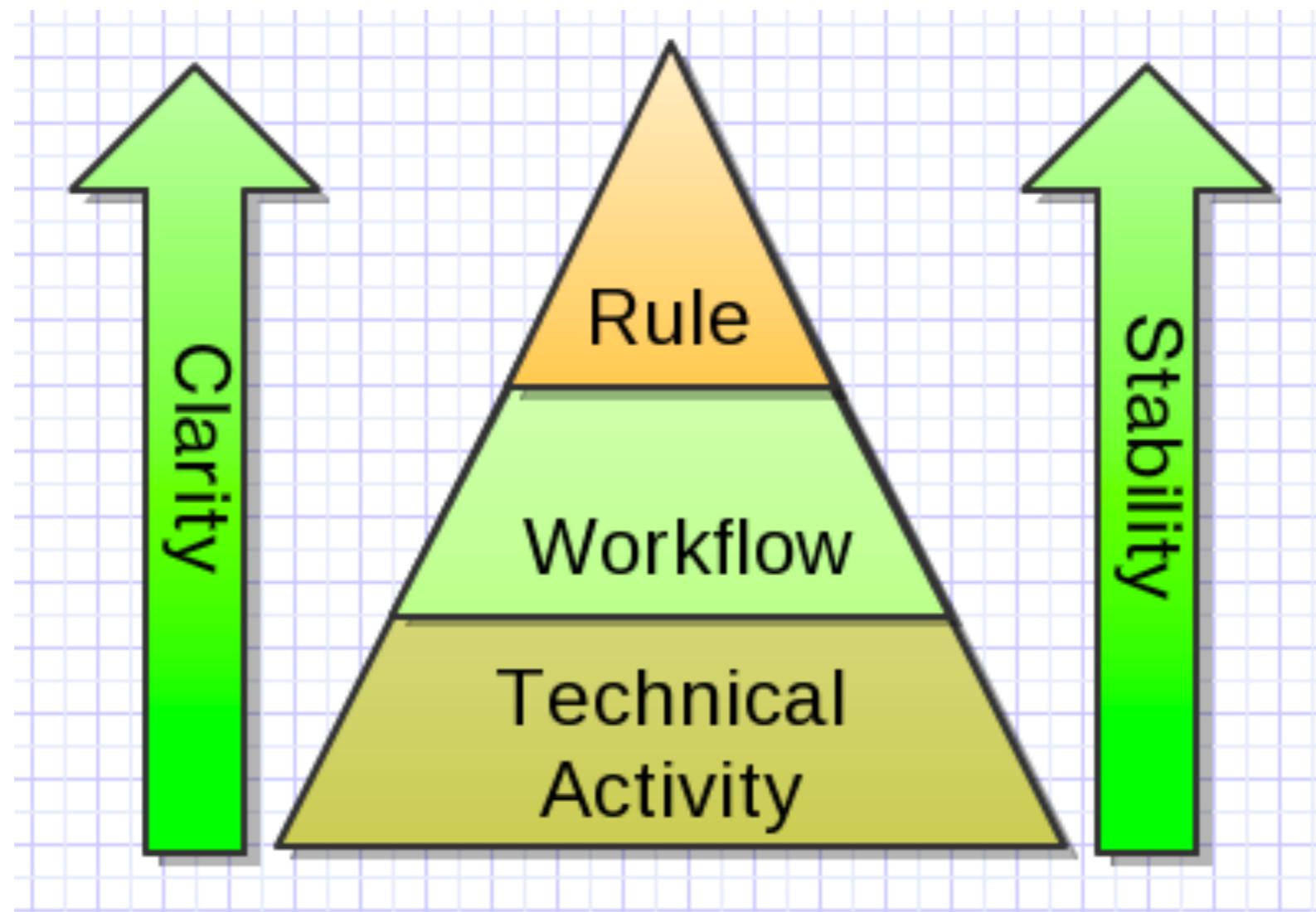
# Integration testing



# Component testing



# 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**

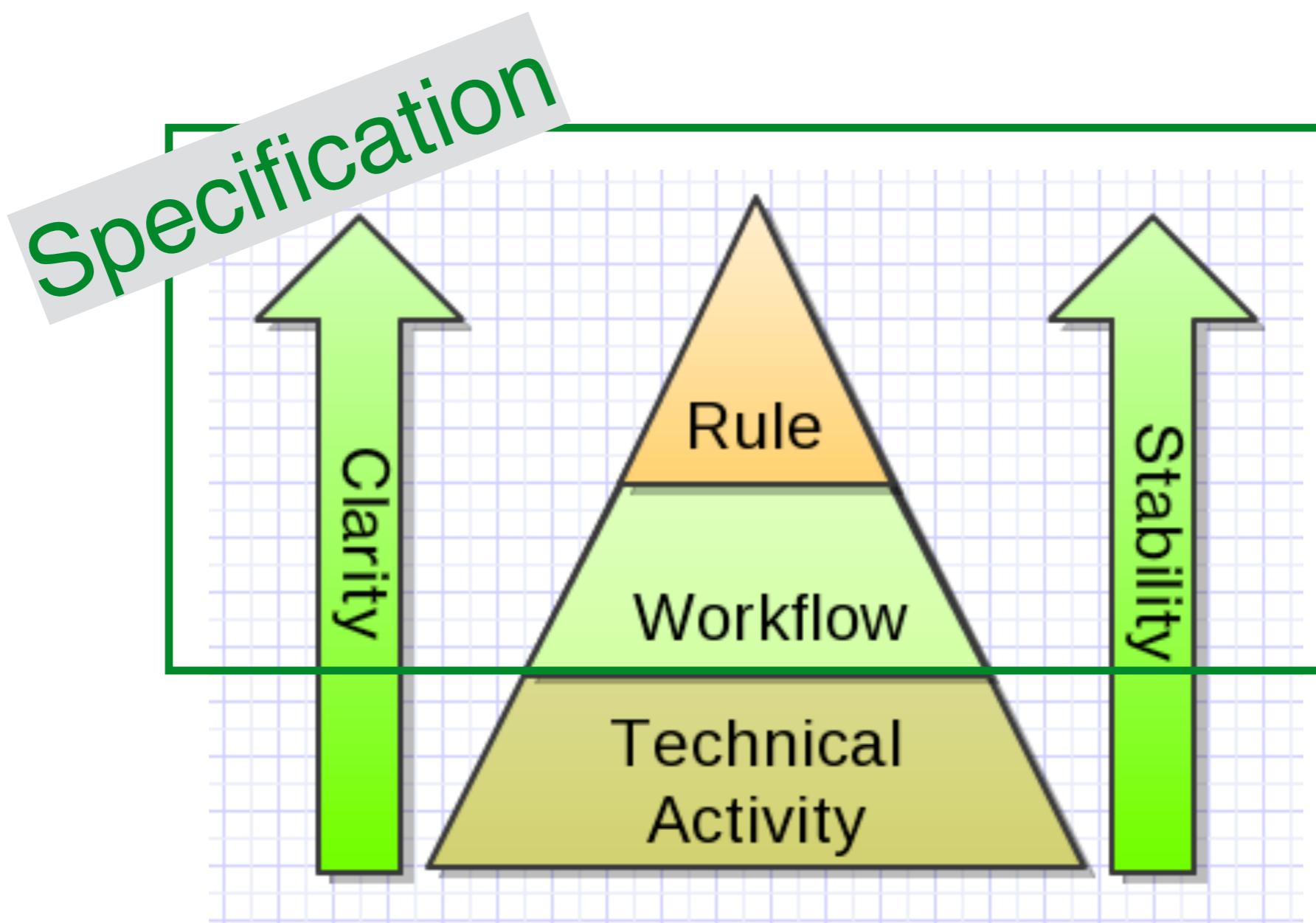
what does a user have to do to exercise the functionality through the UI

## **Technical activity level**

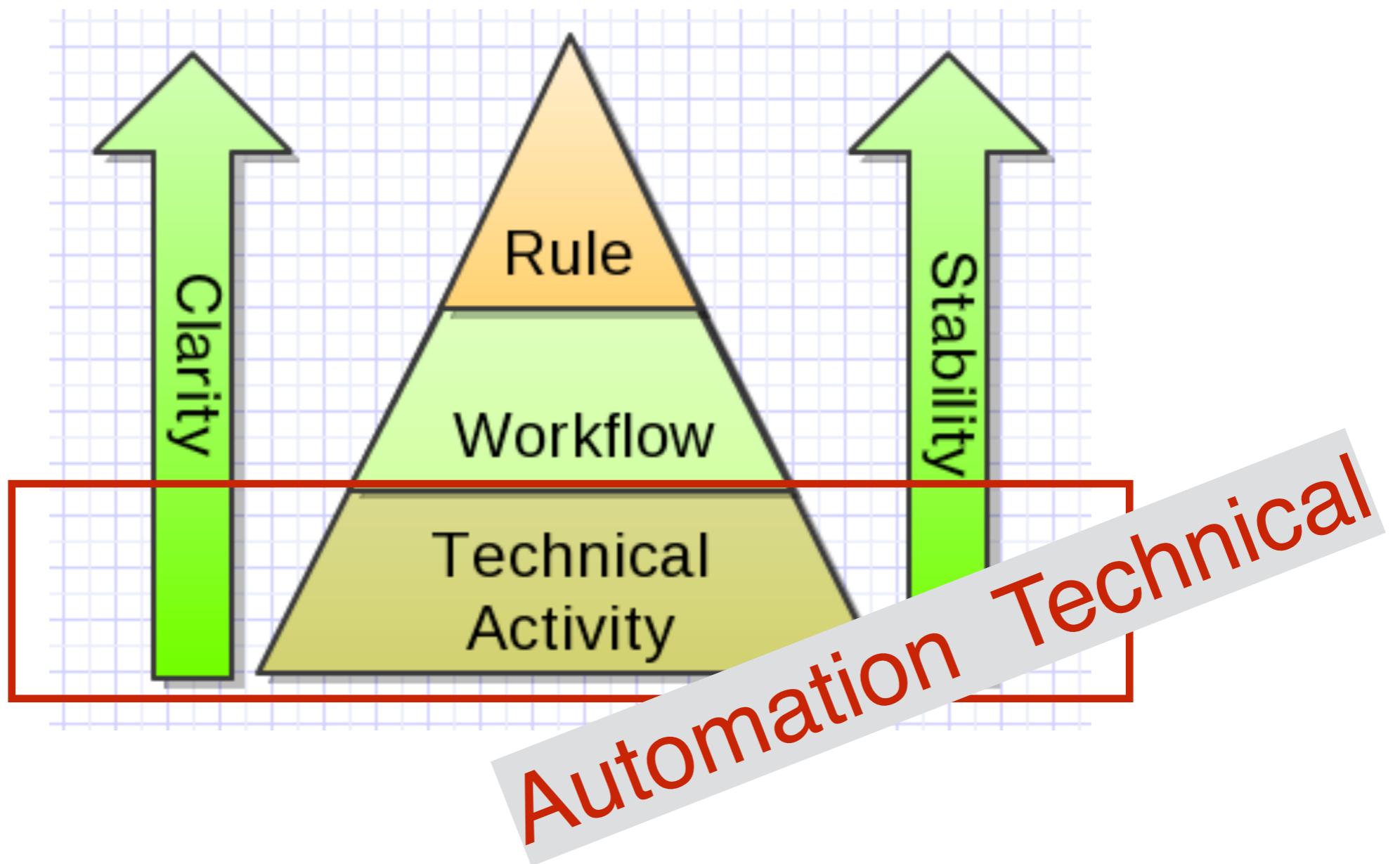
what are the technical steps required to exercise the functionality



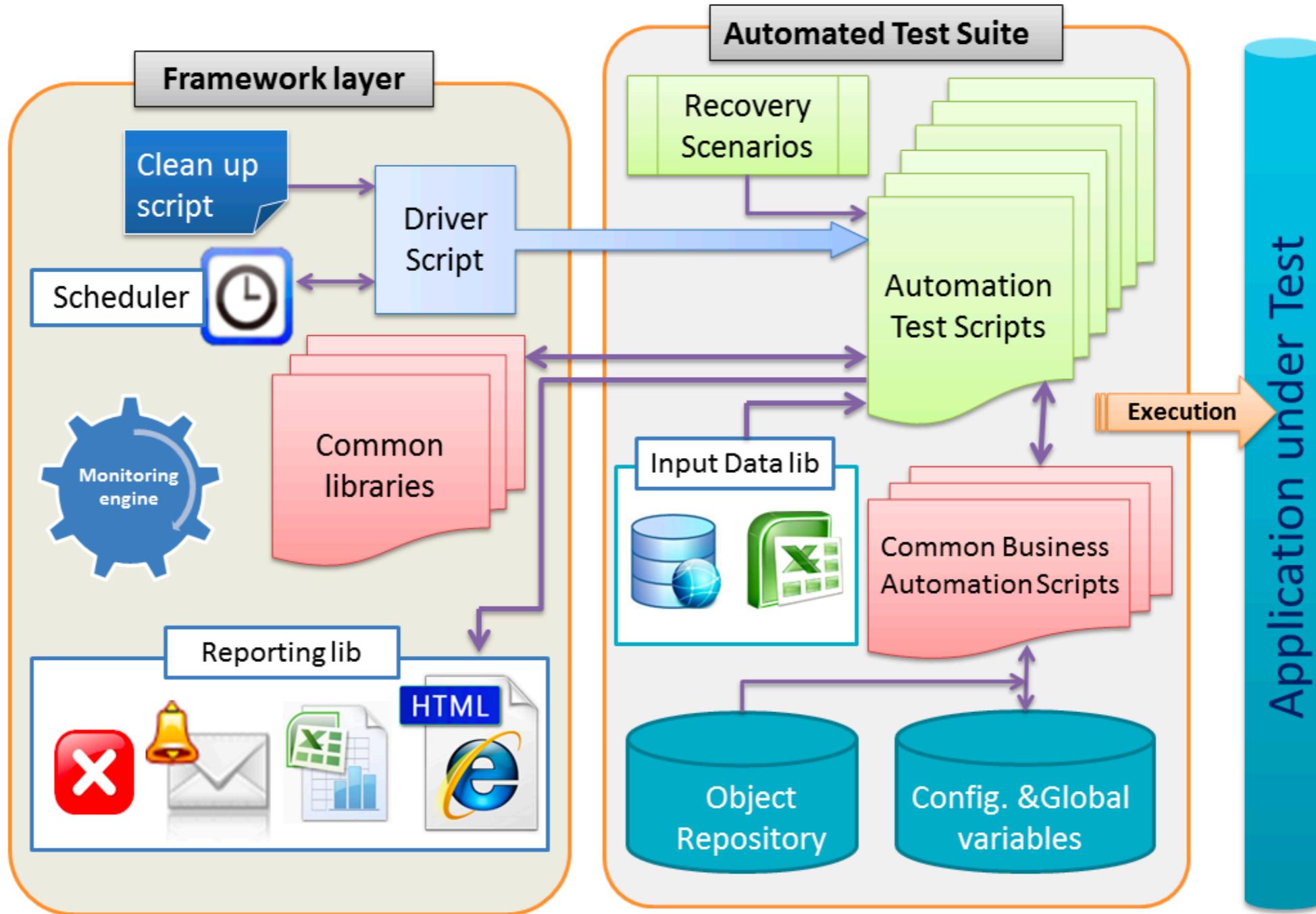
# Good to start



# Good to start



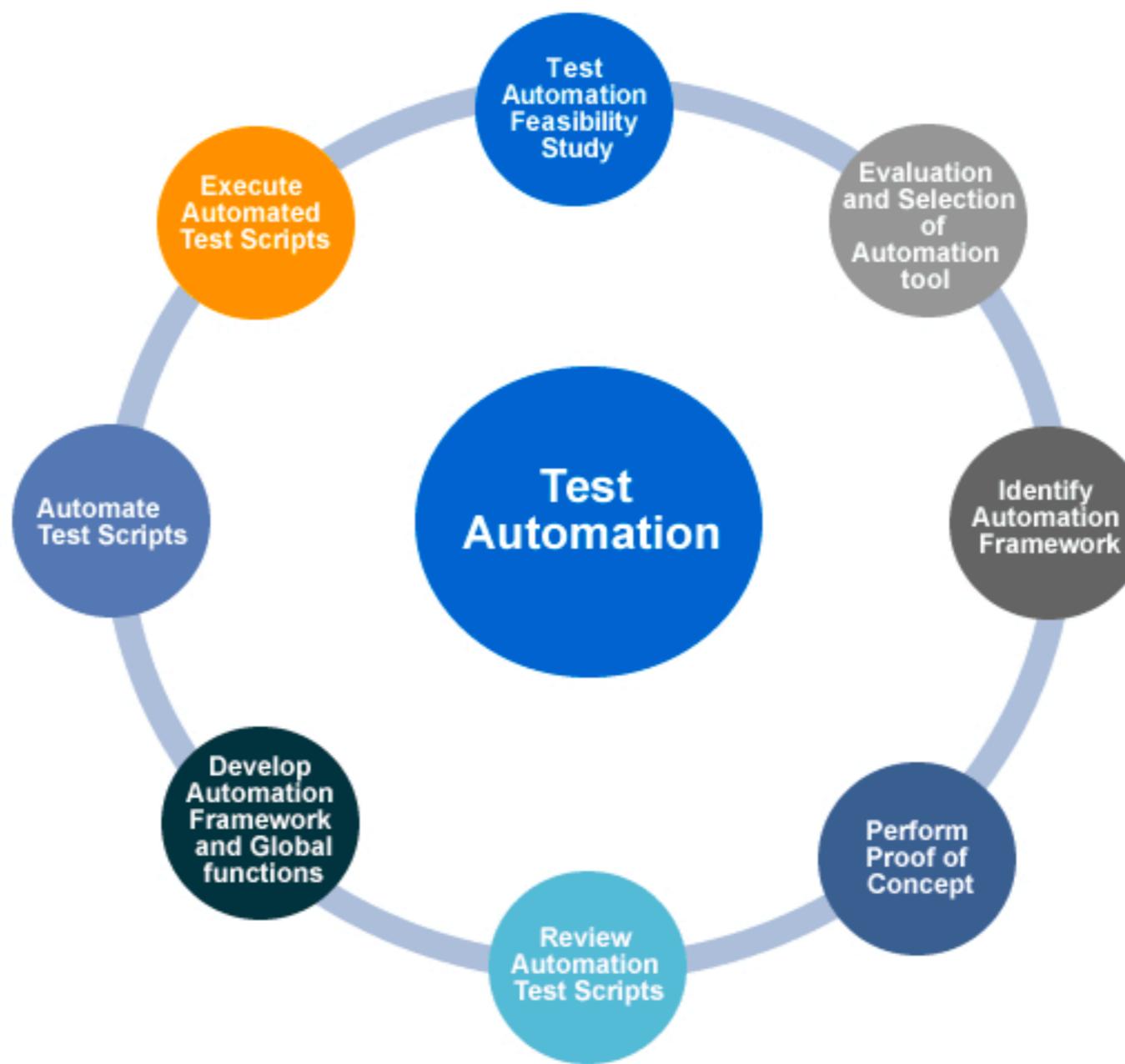
# Test Framework



# Test Automation strategy



# Test Automation selection



# Katalon



<https://www.katalon.com/>



# Overview

Simple and powerful test automation

For web, API and Mobile testing

Easy to start project

Cross platforms support (Windows, mac, Linux)

Full featured to support small to enterprise team



# Katalon features



# Katalon Solutions

## Authoring component

Record and edit

IDE

Keyword

Objects

Data

Test scripts

## Execution component

Test structure

CI tools

Selenium

Appium

Test results

## Report and management

Log and report engine

Test Case Management (TCM)



# Key features

Test case management

Usability and functionality

Test data management and data-driven

Recording and test generation

Build-in keywords

Report and analytics

Integration (Git/CI/JIRA)

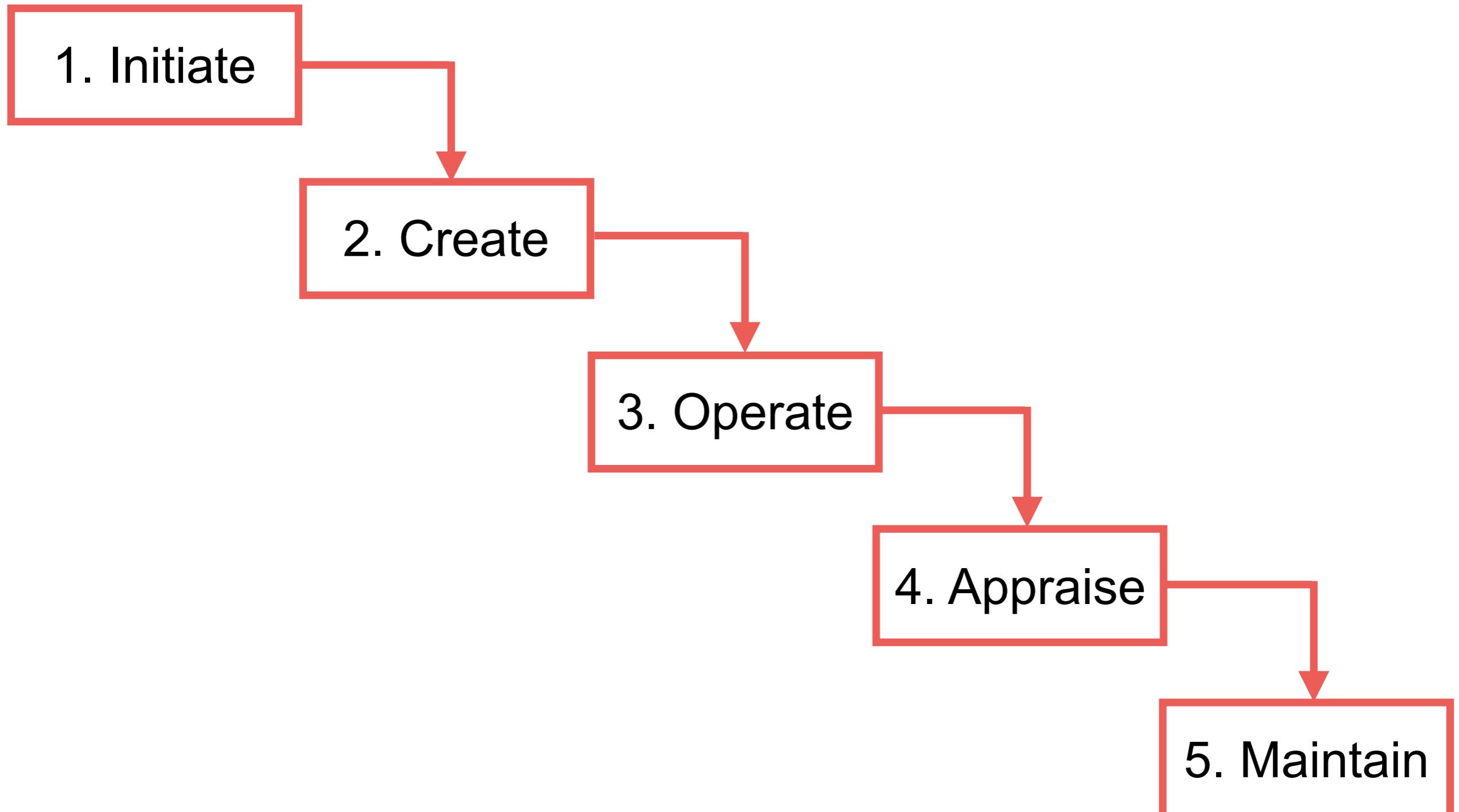


# Automation painpoints

Complexity frameworks  
Experience and expertise  
Reusability of scripts



# Katalon workflow



# 1. Initiate

## Intrinsic project templates

Building templates for organising test case, object, keyword

## Multiple capabilities

Fullt supports web, API and mobile testing

## Hassle-free tool merging

Easy in integrate with Jenkins, Git, JIRA and plug-ins



# 2. Create

## Automatic test generation

Record actions and generate scripts automatically

## Hi-end scripting

Enable building advanced test scripts or customise keywords

## Intuitive object capturing

Advance recorder detects object properties



# 3. Operate

## **Test execution made powerful**

Run test case and test suite with multiple config and data sets

## **Versatility in execution**

Provide CI integration console, remote execution

## **On-the-go failure handling and auto re-execution**

Include runtime rules to handle complex execution flows



# Katalon Integration



# 4. Appraise

**Reports available in several formats**

Advance logging, debug and screenshots

**Bespoke execution reports**

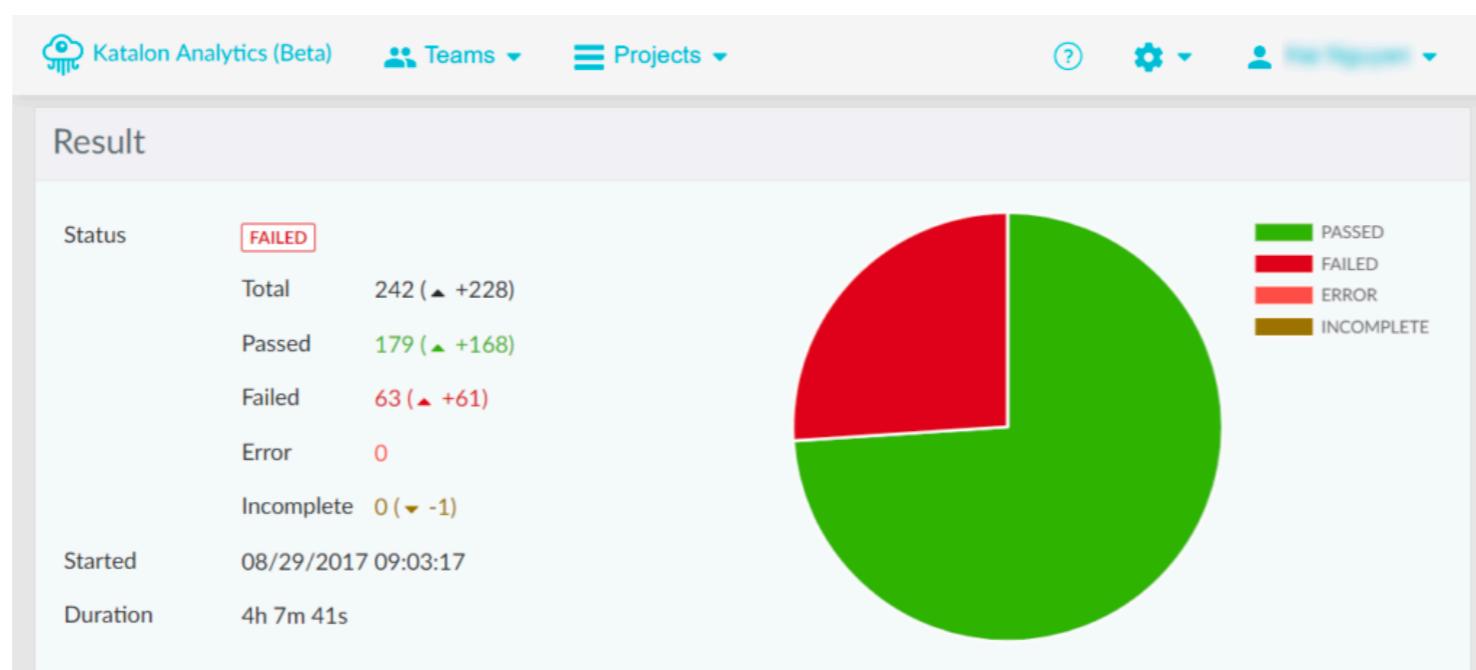
Integrated with your notification workflow

**Enhanced Selenium and Appium logs**

To improve analysis features to improve automation strategy



# Katalon Analytic/Reporting



# 5. Maintain

## **Intelligent test object maintenance**

Auto update all association test cases and suites  
when object changed

## **Efficient test organization**

Make easy to maintain and manage of tests, datas, keywords

## **Easy team collaboration**

Integrate with Git to allow team members  
to share artefacts and workload



# Let's start with Katalon



# Katalon Studio

Java (JDK)  
Android SDK

Web driver for web browsers

Plugins/dependencies

Appium for mobile testing (not included)



# Katalon Studio

Helpful tool for automation teams

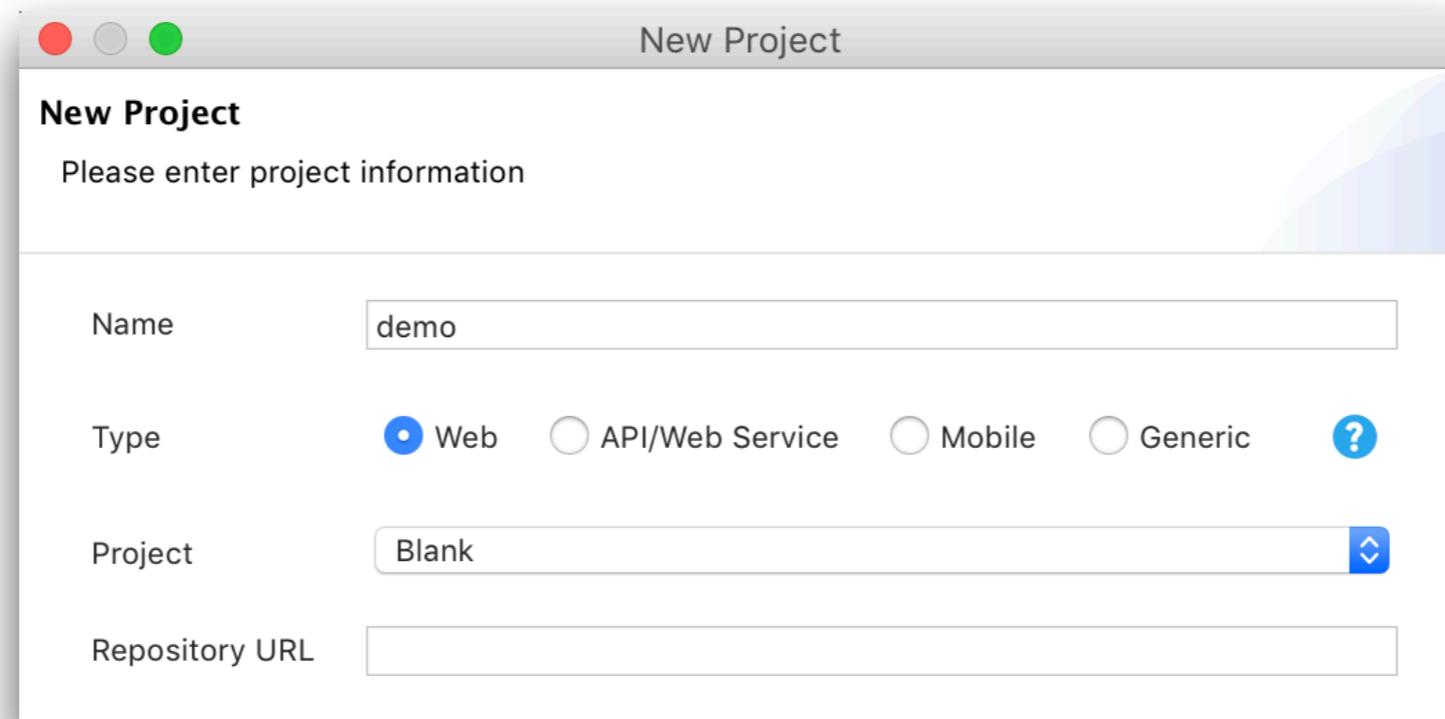
QA/Tester tools

Less effort to setup and execute tests



# Types of project

Web  
API/Web Service  
Mobile  
Generic

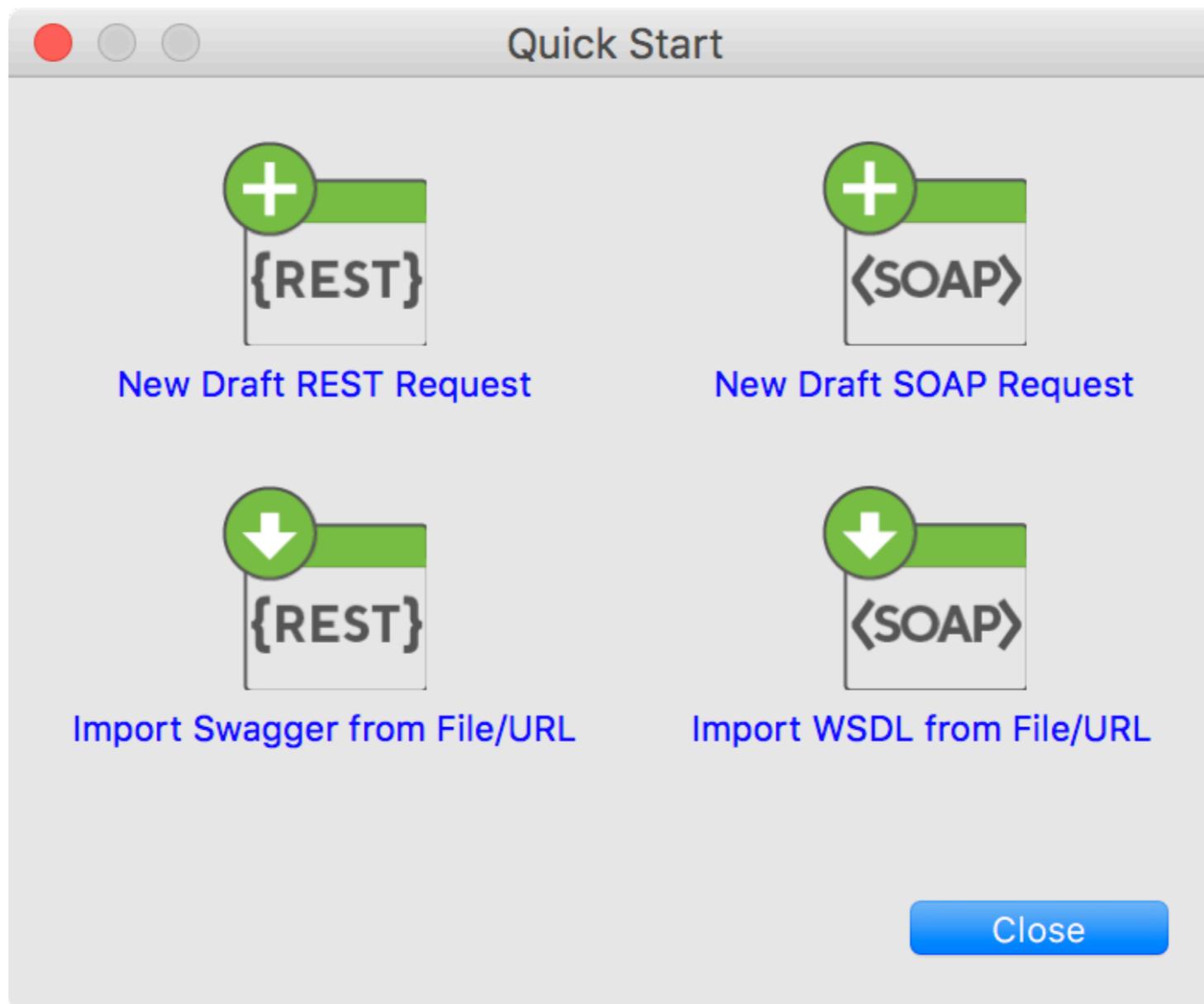


# Web

Record web UI test cases  
Spy objects  
Scripting

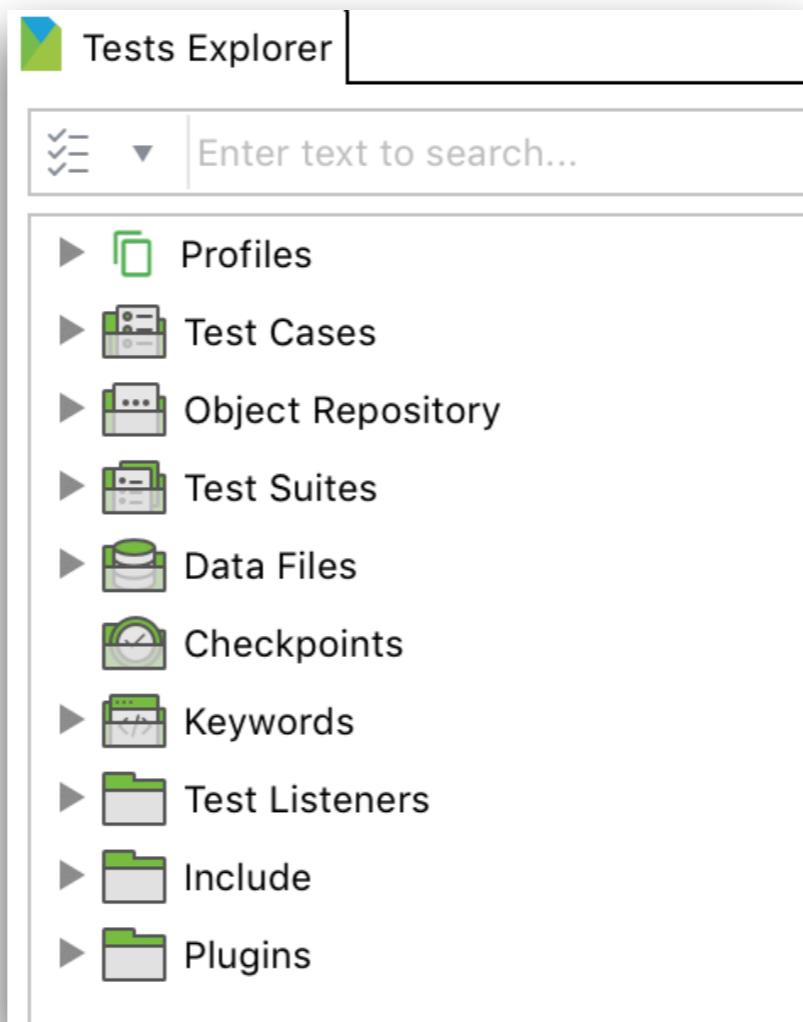


# API/Web Service



# Tests Explorer

Allow to browse the structure of projects  
Access all artefacts quickly

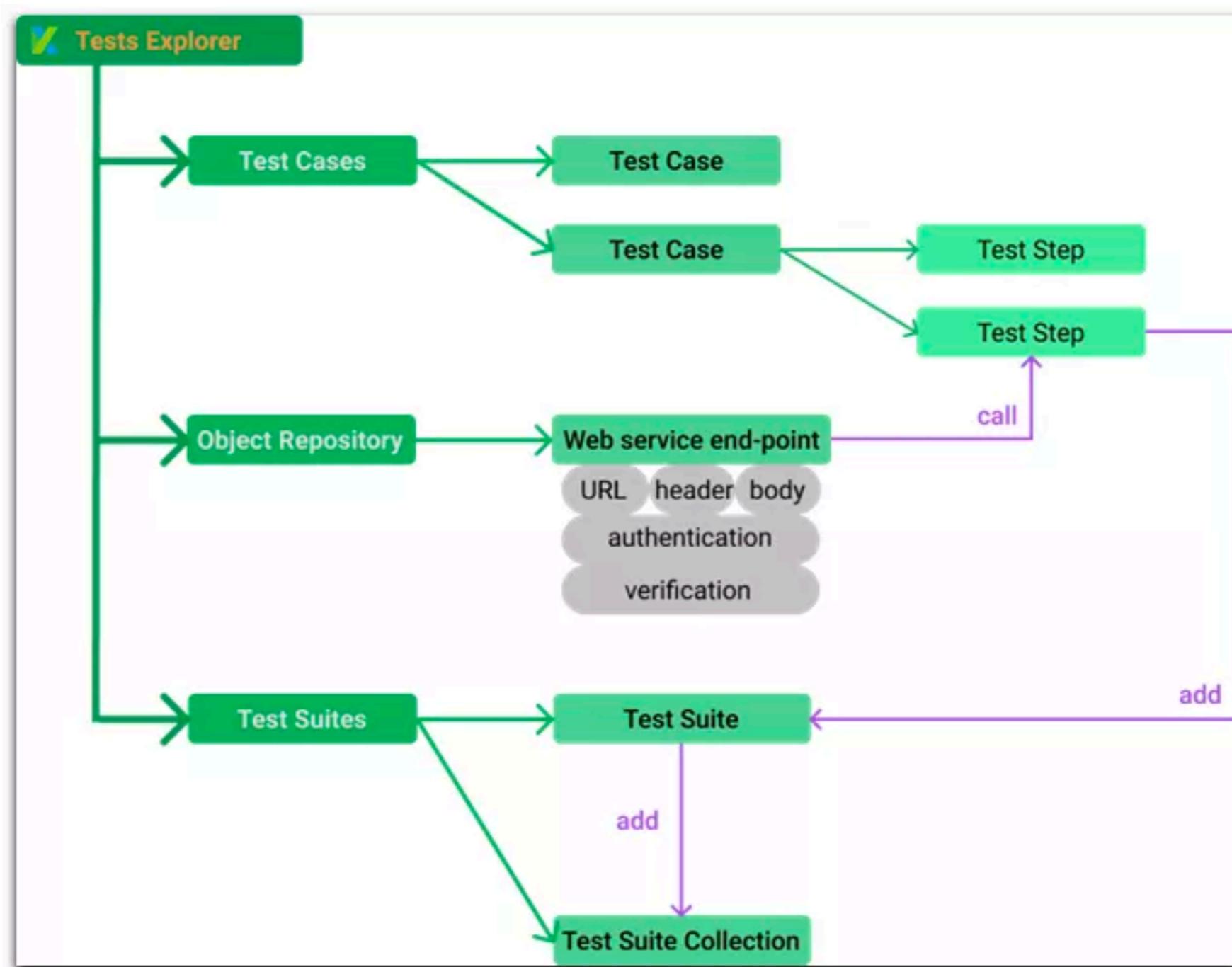


# Vocabulary of Katalon

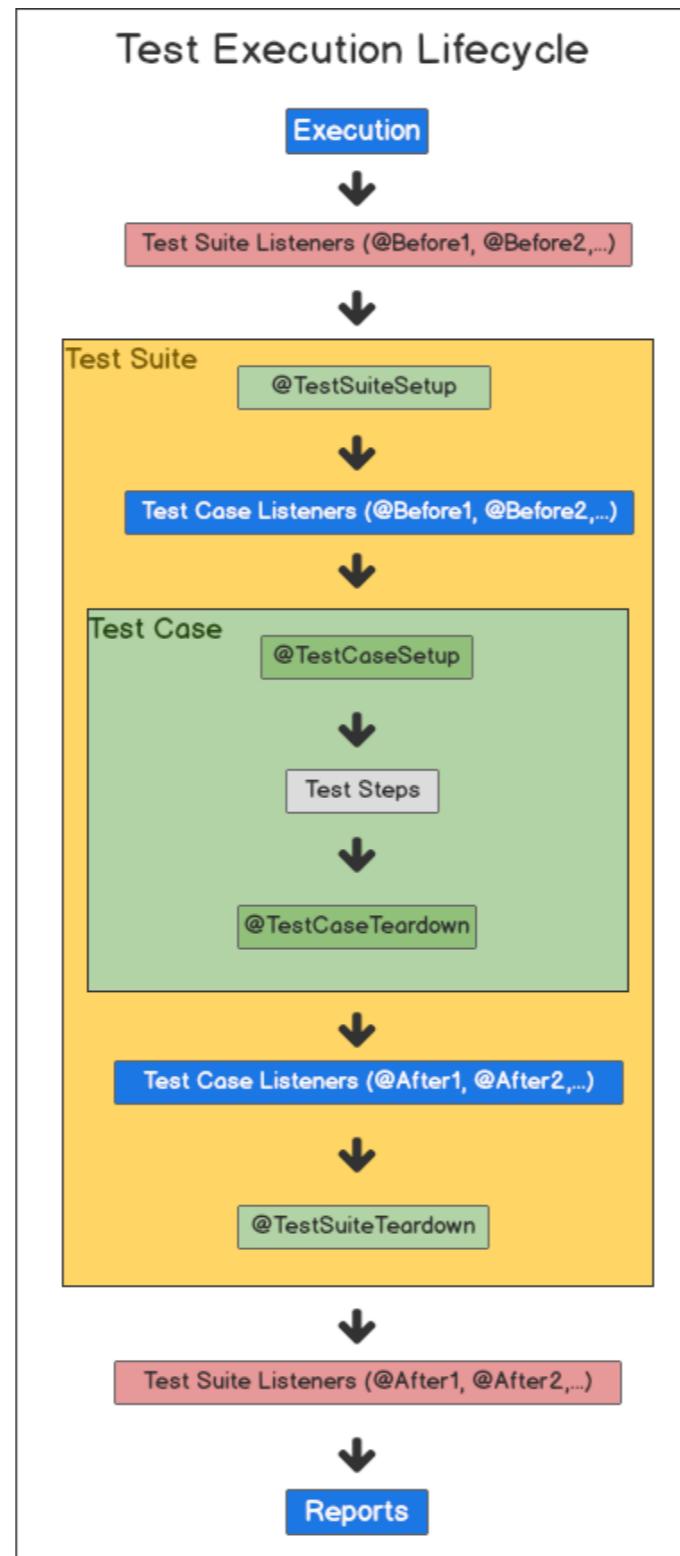
Test cases  
Object repository  
Test suites  
Data files  
Checkpoints  
Keywords  
Test listeners (life cycle)  
Includes  
Plugins



# Project structure

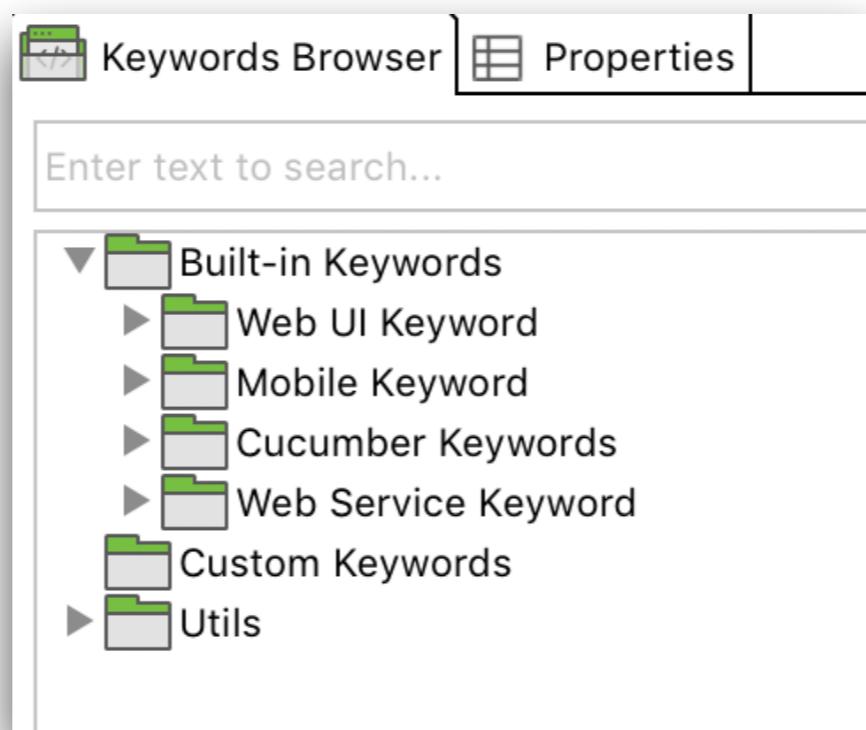


# Test Life Cycle



# Keywords browser

Display all available keywords support by Katalon  
Drag and drop the keywords to test case editor



# Test case editor

The screenshot shows a software interface for test case management. On the left, there's a sidebar with a tree view:

- Tests Explorer (selected)
- Profiles
- Test Cases (selected): Sample
- Object Repository
- Test Suites
- Data Files
- Checkpoints
- Keywords
- Test Listeners
- Include
- Plugins

Below the sidebar is a Keywords Browser window:

- Keywords Browser (selected)
- Properties
- Enter text to search...
- Built-in Keywords
  - Web UI Keyword
  - Mobile Keyword
  - Cucumber Keywords
  - Web Service Keyword
  - Custom Keywords
  - Utils

The main content area, highlighted with a red border, is titled "Sample". It contains a table with four columns: Item, Object, Input, and Output. Below the table are several buttons: Add, Recent keywords, Delete, Move up, Move down, Edit tags, Add to test suite, and View Execution History. At the bottom of the main area are tabs: Manual, </> Script, Variables, </> Variables (Script mode), Integration, and Properties.



# Test case editor

Manual tab

Script tab

Variables tab

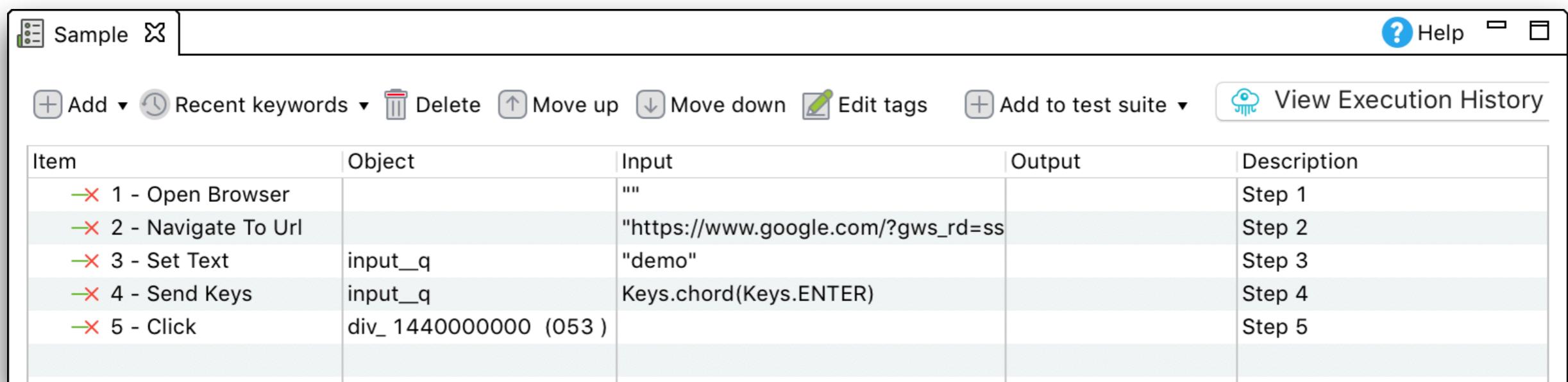
Integration tab

Properties tab



# Manual tab

Display the manual view (basic keyword)  
Allow user to create automation test effortlessly



The screenshot shows a software application window titled "Sample". The main area contains a table with five rows, each representing a step in a test case. The columns are labeled "Item", "Object", "Input", "Output", and "Description". The steps are numbered 1 through 5 and describe actions like opening a browser, navigating to a URL, setting text in an input field, sending keys, and clicking a specific element.

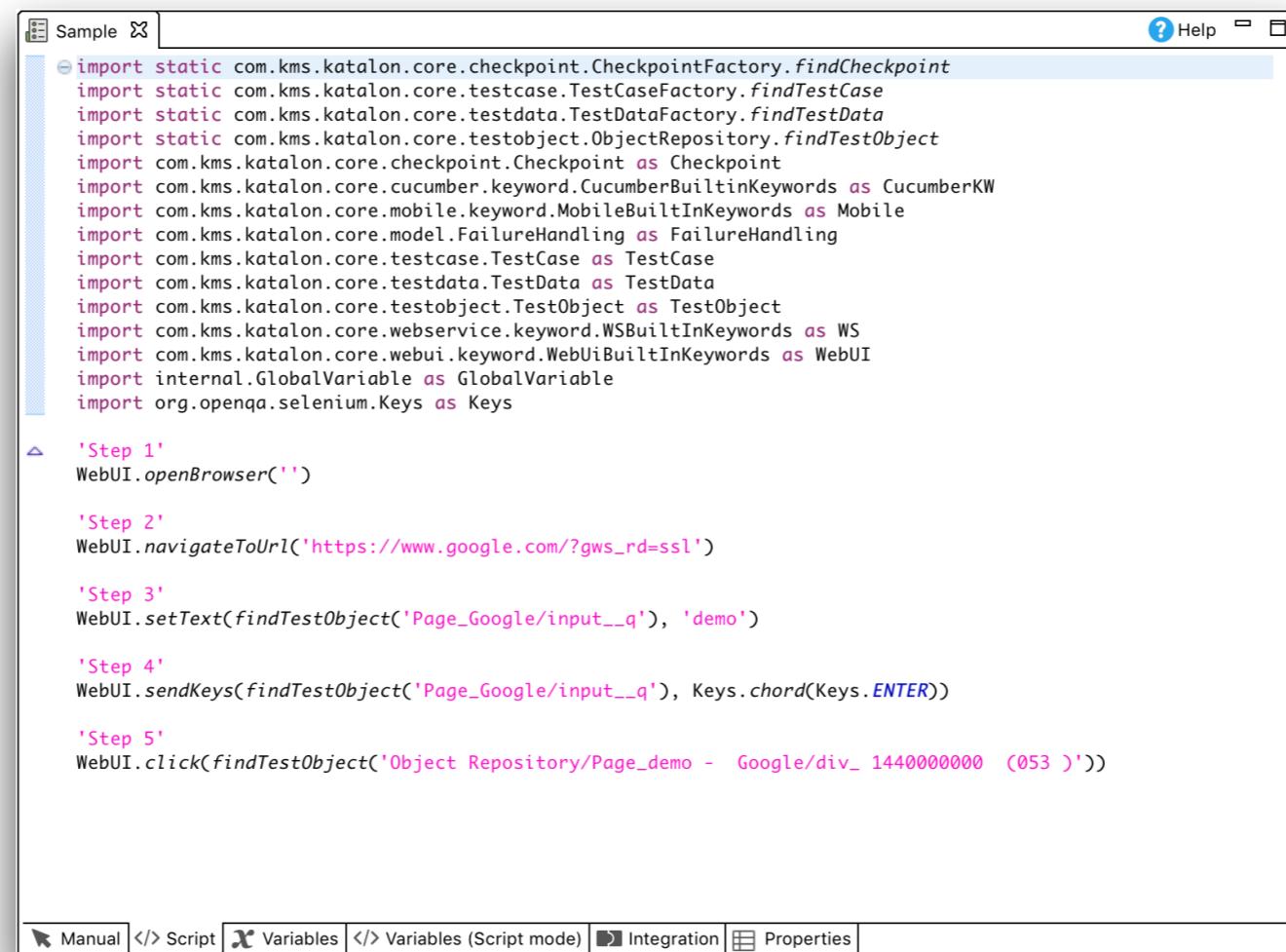
Item	Object	Input	Output	Description
1 - Open Browser		""		Step 1
2 - Navigate To Url		"https://www.google.com/?gws_rd=ss		Step 2
3 - Set Text	input_q	"demo"		Step 3
4 - Send Keys	input_q	Keys.chord(Keys.ENTER)		Step 4
5 - Click	div_1440000000 (053 )			Step 5



# Script tab

Display the script view

Allow advance user to write code in test case  
Write code with Java or Groovy



The screenshot shows the 'Sample' test case in Katalon Studio's Script tab. The code is written in Groovy and defines a series of steps to open a browser, navigate to Google, search for 'demo', and click a specific result. The code uses various Katalon core and web UI keywords.

```
import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint
import static com.kms.katalon.core.testcase.TestCaseFactory.findTestCase
import static com.kms.katalon.coretestdata.TestDataFactory.findTestData
import static com.kms.katalon.core.testobject.ObjectRepository.findTestObject
import com.kms.katalon.core.checkpoint.Checkpoint as Checkpoint
import com.kms.katalon.core.cucumber.keyword.CucumberBuiltInKeywords as CucumberKW
import com.kms.katalon.core.mobile.keyword.MobileBuiltInKeywords as Mobile
import com.kms.katalon.core.model.FailureHandling as FailureHandling
import com.kms.katalon.core.testcase.TestCase as TestCase
import com.kms.katalon.coretestdata.TestData as TestData
import com.kms.katalon.core.testobject.TestObject as TestObject
import com.kms.katalon.core.webservice.keyword.WSBuiltInKeywords as WS
import com.kms.katalon.core.webui.keyword.WebUiBuiltInKeywords as WebUI
import internal.GlobalVariable as GlobalVariable
import org.openqa.selenium.Keys as Keys

△ 'Step 1'
WebUI.openBrowser('')

'Step 2'
WebUI.navigateToUrl('https://www.google.com/?gws_rd=ssl')

'Step 3'
WebUI.setText(findTestObject('Page_Google/input__q'), 'demo')

'Step 4'
WebUI.sendKeys(findTestObject('Page_Google/input__q'), Keys.chord(Keys.ENTER))

'Step 5'
WebUI.click(findTestObject('Object Repository/Page_demo - Google/div_ 1440000000 (053)'))
```



# Variables tab

Show all variables defined for test case  
Test case/ Groovy/ Global variables

A screenshot of a software interface titled "Sample". At the top, there are buttons for "Add", "Delete", "Clear", "Move up", and "Move down". Below is a table with columns: No., Name, Type, and Default value. One row is present in the table.

No.	Name	Type	Default value
1	keyword	String	"demo"



# Properties tab

## General information about test cases

Sample X

Help

ID	Test Cases/Sample	Name	Sample
Created Date	Mon Oct 21 09:02:27 ICT 2019	Modified Date	Mon Oct 21 22:48:40 ICT 2019
Tag		Comment	
Description			



# User Interface Testing

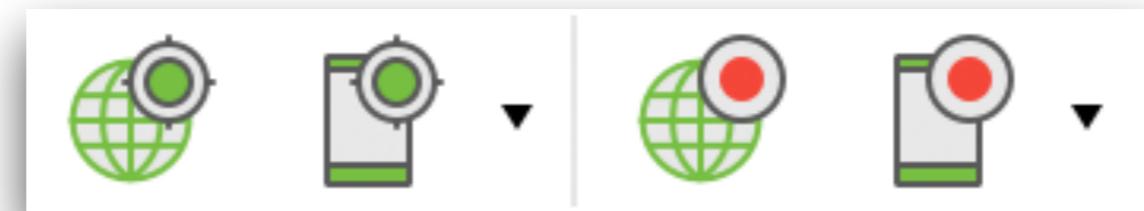


# Working with web browser

Spy web  
Web recording  
Run scripts  
Report  
Custom scripts



# Toolbars

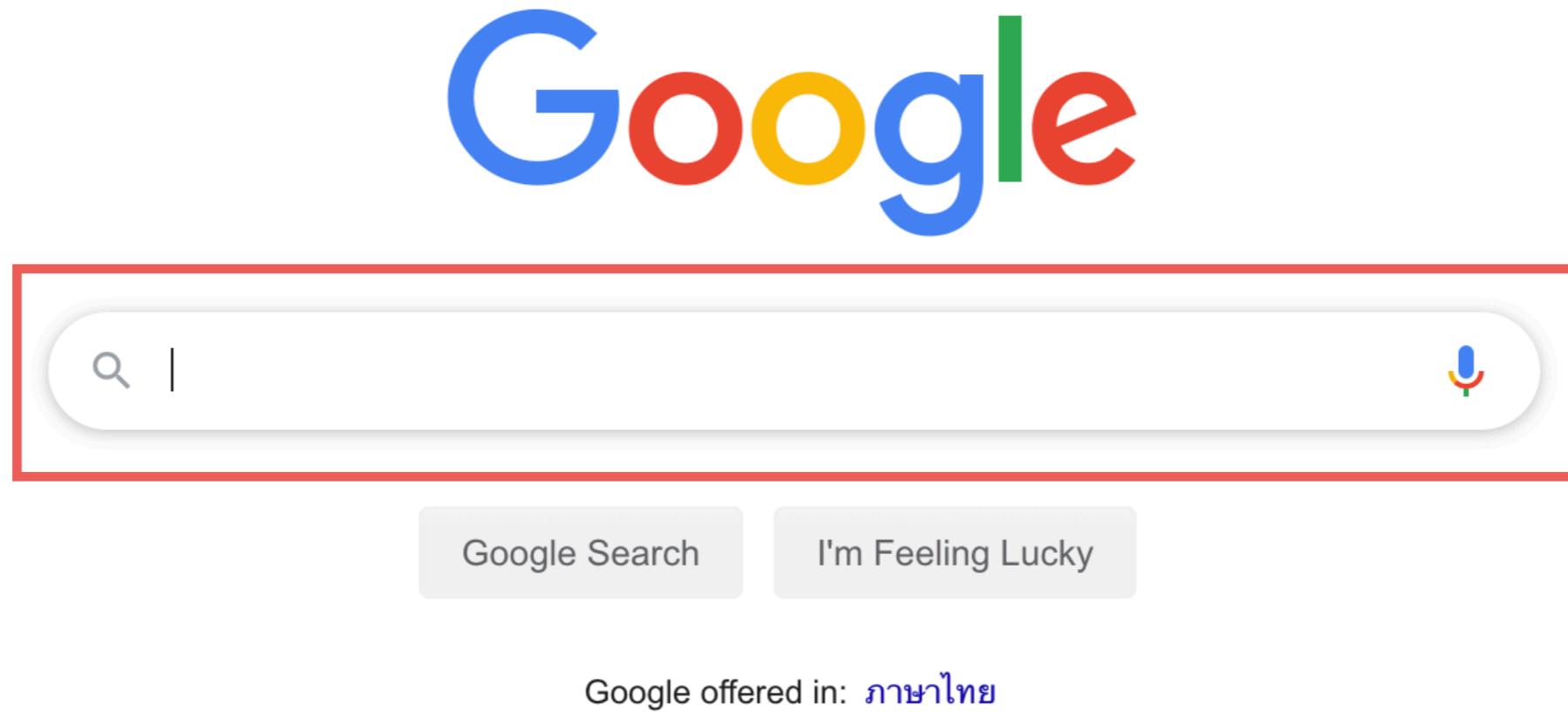


Spy web

Record web



# How to access element ?



# Spy web

Capture objects from web browser  
Save objects to object repository



# Spy web (1)

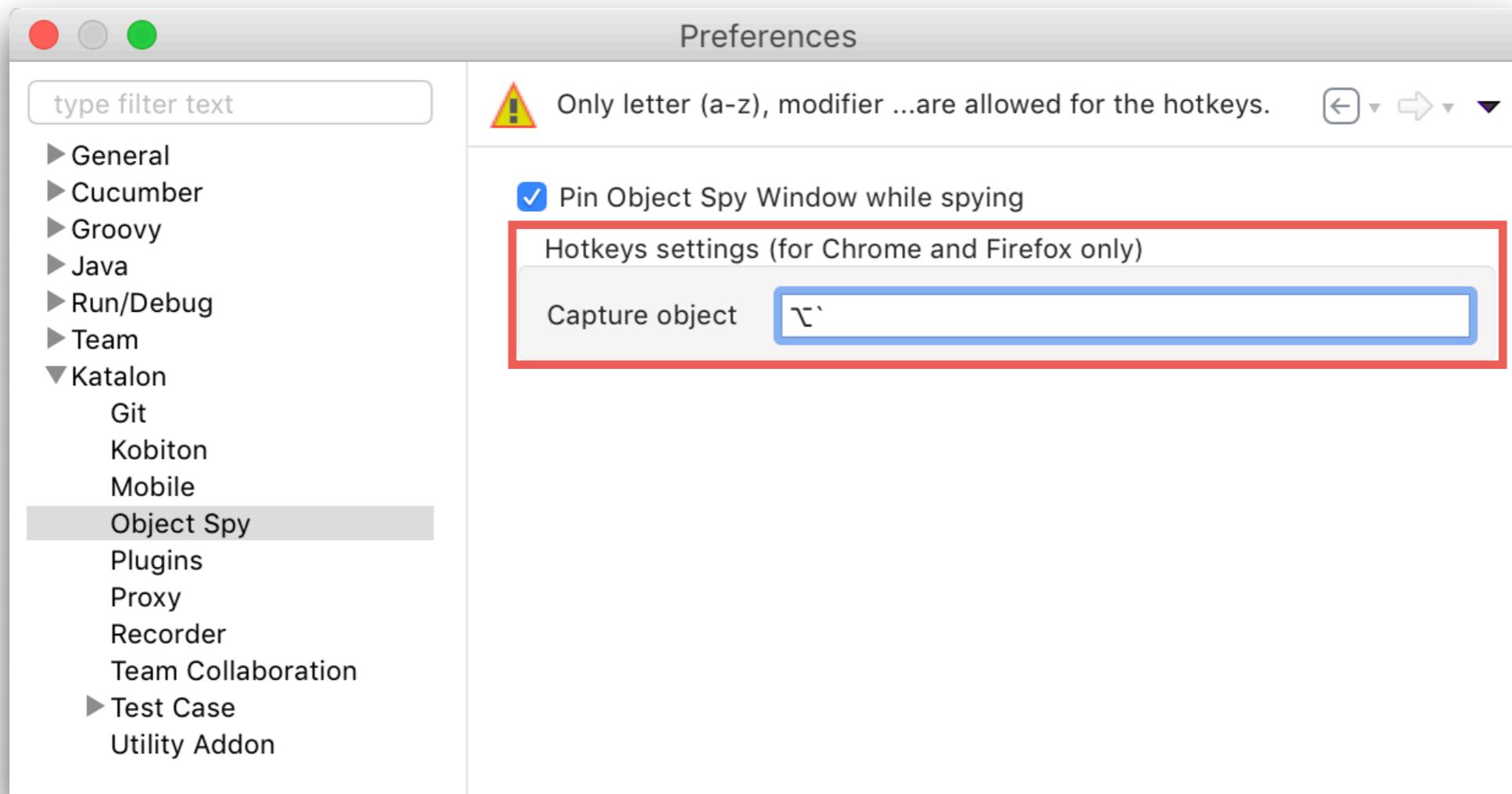
The screenshot shows a Google search page in a browser window. The Google logo is highlighted with a red rectangular box. To the right of the browser is the 'Object Spy' extension interface. The 'CAPTURED OBJECTS' section is expanded, showing a tree view with 'Page\_Google' expanded to show 'img\_hplogo', 'input\_q', 'input\_btnK', and 'input\_btnL'. The 'OBJECT PROPERTIES' and 'Selected Locator' sections are also visible.

Capture object Alt + `

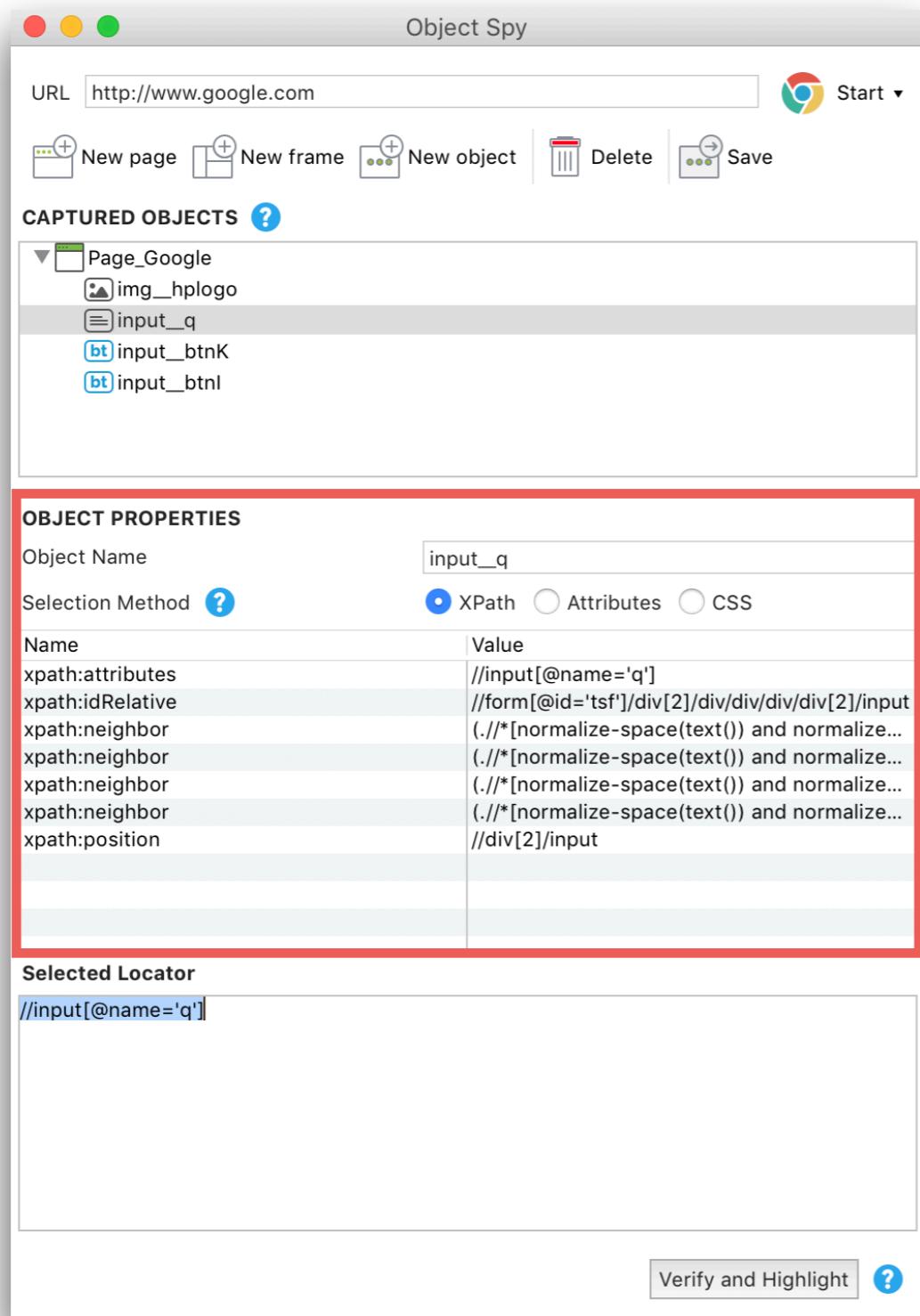


# Change key to capture object

Preferences -> Katalon -> Object spy



# Spy web (2)



Object  
properties



# Spy web (3)

Object Spy

URL <http://www.google.com> Start ▾

New page New frame New object Delete Save

CAPTURED OBJECTS ?

- Page\_Google
  - img\_hplogo
  - input\_q
  - input\_btnK
  - input\_btnL

OBJECT PROPERTIES

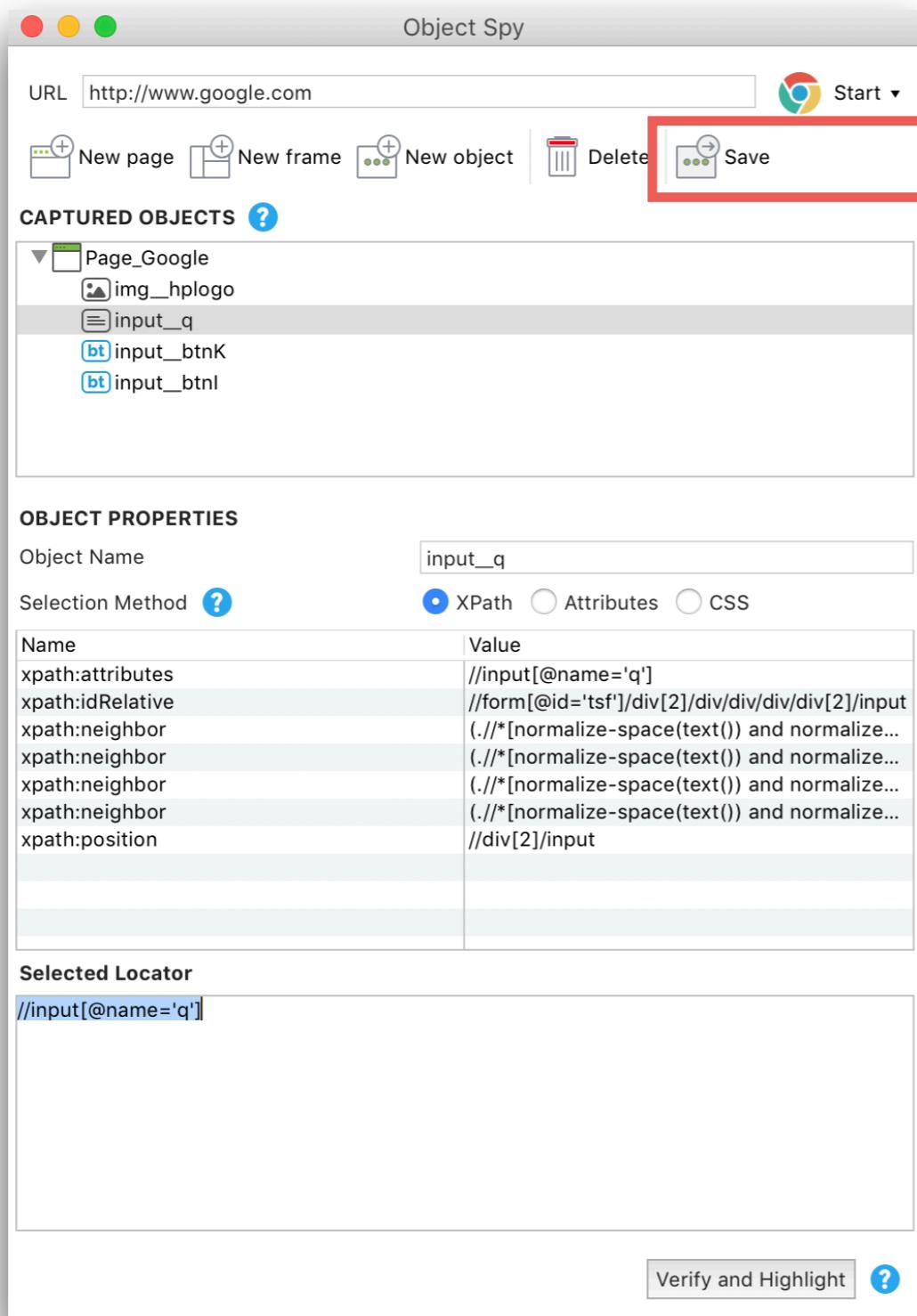
Object Name input\_q

Selection Method  XPath  Attributes  CSS

Name	Value
xpath:attributes	//input[@name='q']
xpath:idRelative	//form[@id='tsf']/div[2]/div/div/div[2]/input
xpath:neighbor	(.//*[normalize-space(text()) and normalize...]
xpath:position	//div[2]/input

Selected Locator  
`//input[@name='q']`

Verify and Highlight ?

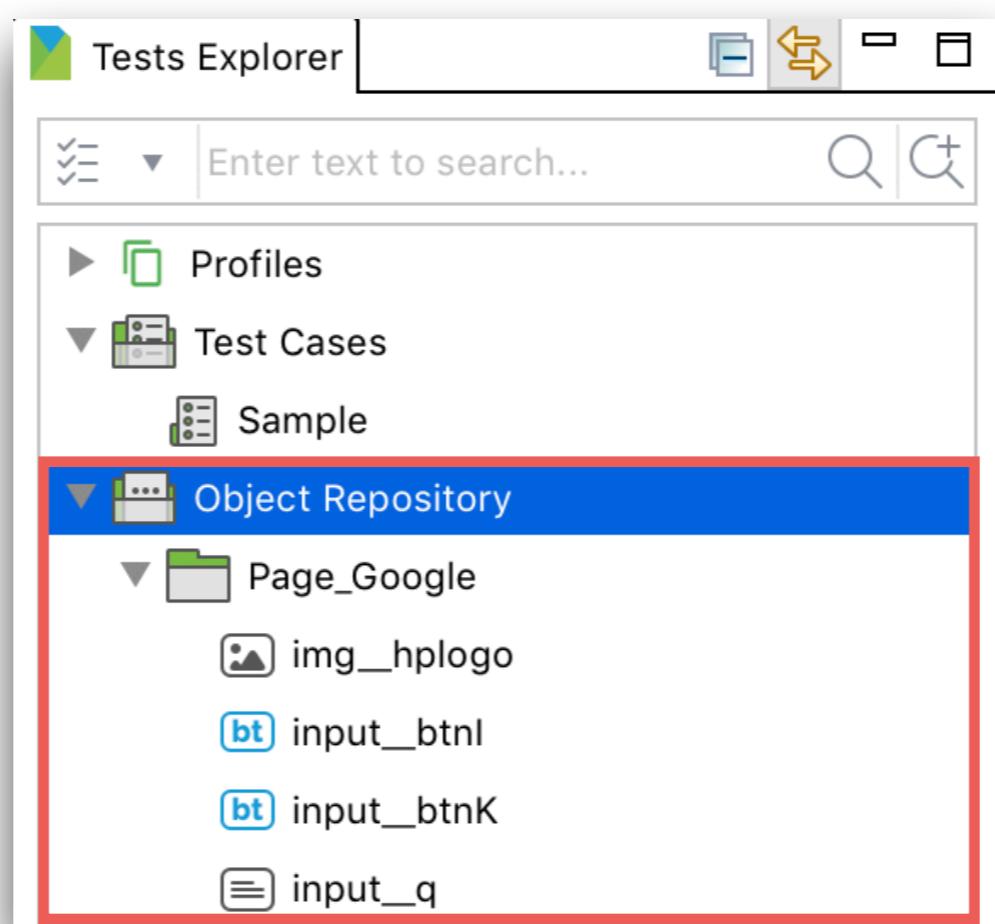


→ Save to object repository



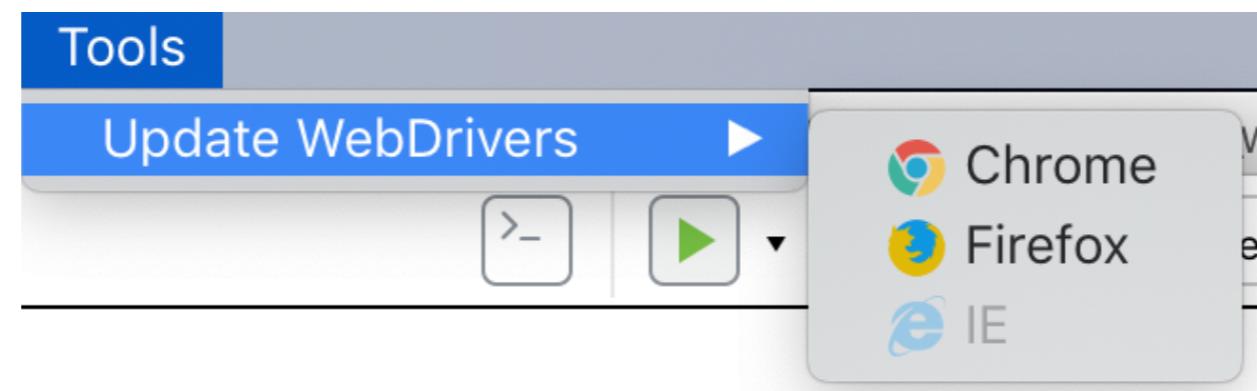
# Spy web (4)

Save objects in folder/page object



# Tips

Always Update Browser Driver !!



# Element locators



# Locator

A way to find a web element by selenium  
Locator strategies (id, name, xpath, etc.)



# Locators

Locator	Description
Id	Select element with the specified @id attribute
Name	Select first element with the specified @name attribute
Link text	Select link element which contains text matching the specified link text
Partial link text	Select link element which contains text matching the specified partial link text
Tag name	Locate Element using a Tag name
Class name	Locate Element using a class name
CSS	Select the element using CSS selectors
Xpath	Locate an element using a XPath expression



# Challenges

Unable to find elements  
Multiple elements matched  
Hard to maintain scripts  
Change in implementation

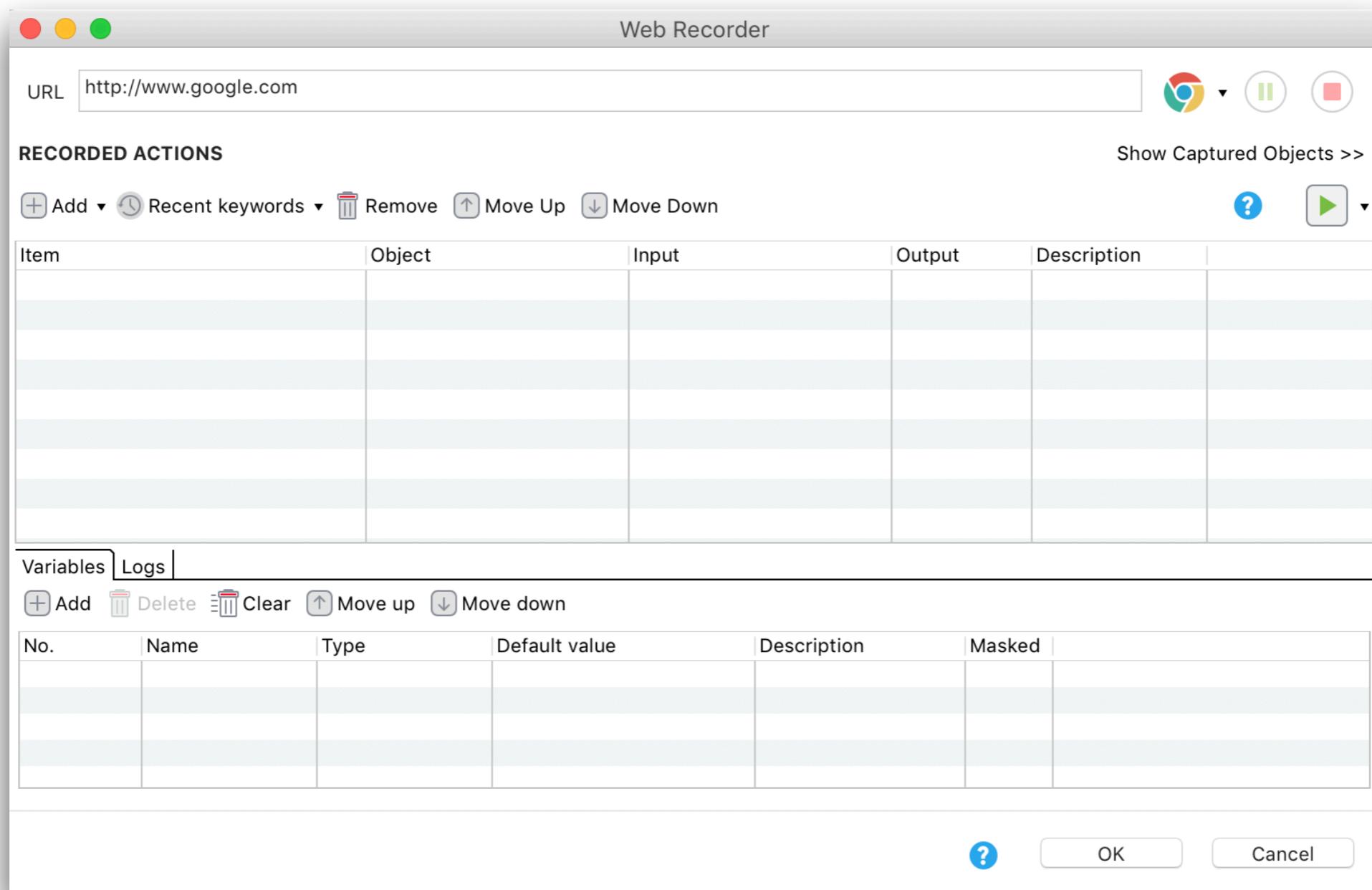


# Web recording

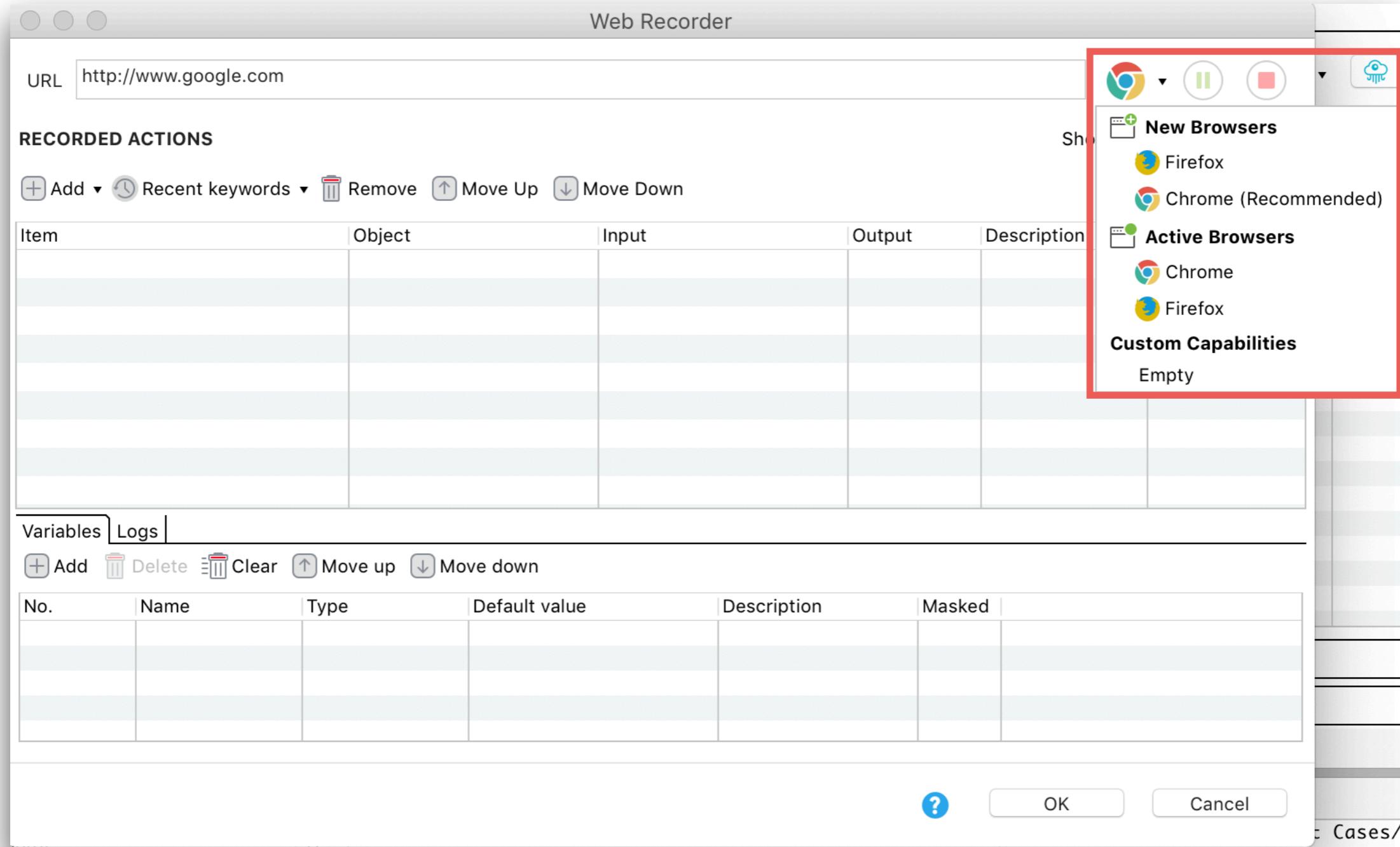


# Web recording

Record all actions from web browser



# Try to record (choose browser)



# Let's record

Web Recorder

URL  Record ▾   

**RECORDED ACTIONS** Show Captured Objects >>

 Add ▾  Recent keywords ▾  Remove  Move Up 

Item	Object	Input	Output	Description
1 - Open Browser		""		
2 - Navigate To Url		"https://www.google.com/?g		
3 - Set Text	input_q	"demo"		
4 - Send Keys	input_q	Keys.chord(Keys.ENTER)		
5 - Close Browser				

 Variables  Logs

 Add  Delete  Clear  Move up  Move down

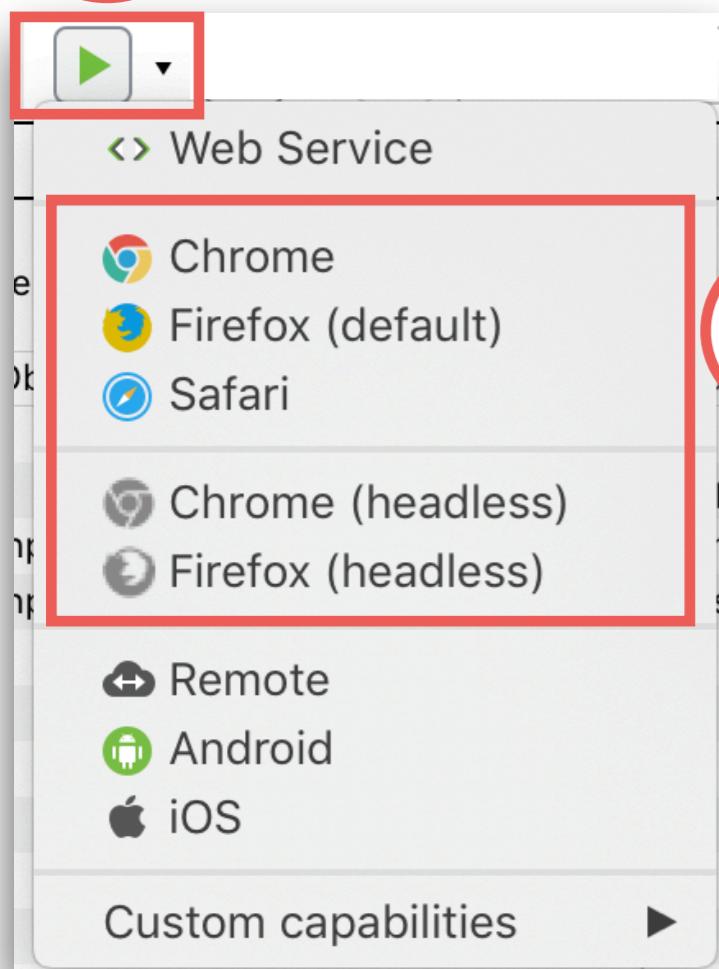
No.	Name	Type	Default value	Description	Masked

 OK Cancel



# Run test cases

1 Click run test case



2 Choose target/browser to run tests

3 Test results

Job Progress	
●	Test Cases/Demo2 - Firefox - 20191021_231100 1/1
✓	<Done> - Firefox
●	Test Cases/Demo2 - Chrome - 20191021_231038 1/1
✓	<Done> - Chrome



# Test logs

The screenshot shows the Katalon Studio interface with the 'Log Viewer' tab selected. At the top, it displays statistics: Runs: 1/1, Passes: 1, Failures: 0, Errors: 0, and Skips: 0. Below this, a tree view shows a test suite named 'Test Cases/Demo2' which took 6.584s. The suite contains five test steps: 1 - openBrowser("") (1.323s), 2 - navigateToUrl("https://www.google.com/?gws\_rd=ssl") (1.609s), 3 - setText(findTestObject("Page\_Google/input\_q"), "demo") (0.204s), 4 - sendKeys(findTestObject("Page\_Google/input\_q"), Keys.chord(ENTER)) (3.124s), and 5 - closeBrowser(STOP\_ON\_FAILURE) (0.092s). All steps are marked with green checkmarks, indicating they passed.

<https://docs.katalon.com/katalon-studio/docs/test-suite-report.html>



# More reports

Create new test suite -> add test cases

The screenshot shows the Katalon Studio interface. On the left, the 'Tests Explorer' sidebar lists various project components: Profiles, Test Cases, Object Repository, Test Suites (with 'suite\_01' selected), Data Files, Checkpoints, Keywords, and Test Listeners. The 'Test Suites' section is highlighted with a red box. The main window displays the 'suite\_01' test suite. At the top, there are filter checkboxes for Passed, Failed, Error, and Incomplete test cases. Below the filters is a search bar labeled 'Search here...'. A table titled 'Test Cases Table' lists one test case: 'Demo2 (7.461s)'.

No.	Name	Video
1	Demo2 (7.461s)	



# Run suite and see report

The screenshot shows the Katalon Studio interface displaying the results of a test suite named "suite\_01".

**Test Cases Table:**

- Filter buttons: Passed (green checkmark), Failed (red X), Error (red exclamation mark), Incomplete (blue circle).
- Analytics: Katalon Analytics icon.
- Search bar: Search here... with a magnifying glass icon.
- Table columns: No., Name, and Video.
- Table data:

1	Demo2 (7.461s)	
---	----------------	--

**Summary:**

Test Suite ID	<a href="#">Test Suites/suite_01</a>		
Host name	somkiat - 192.168.1.33	OS	Mac OS X 64bit
Katalon version	6.3.3.11	Platform	Chrome 77.0.3865.120
Start	2019-10-21 23:21:08	End	2019-10-21 23:21:16
Elapsed	7.645s		
Total TC	1		
Passed	1	Failed	0
Error	0	Incomplete	0

**Main | Script | Integration | Result**



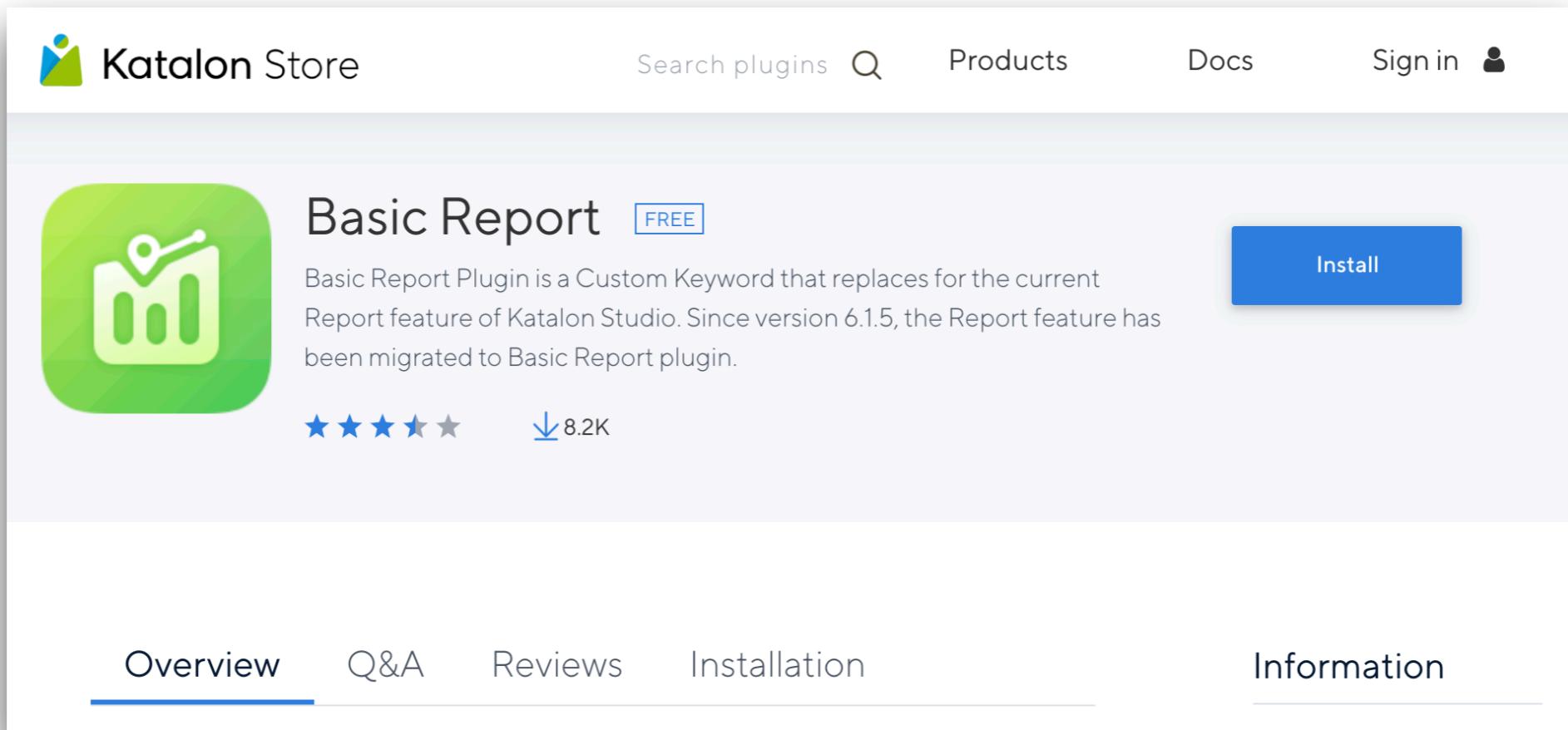
# Export report to other formats

Since Katalon 6.1.5, report feature migrated to  
Basic report plugin  
Support formats (HTML, CSV, Junit, PDF)



# Export report to other formats

Try to install Basic report plug-in and reload !!

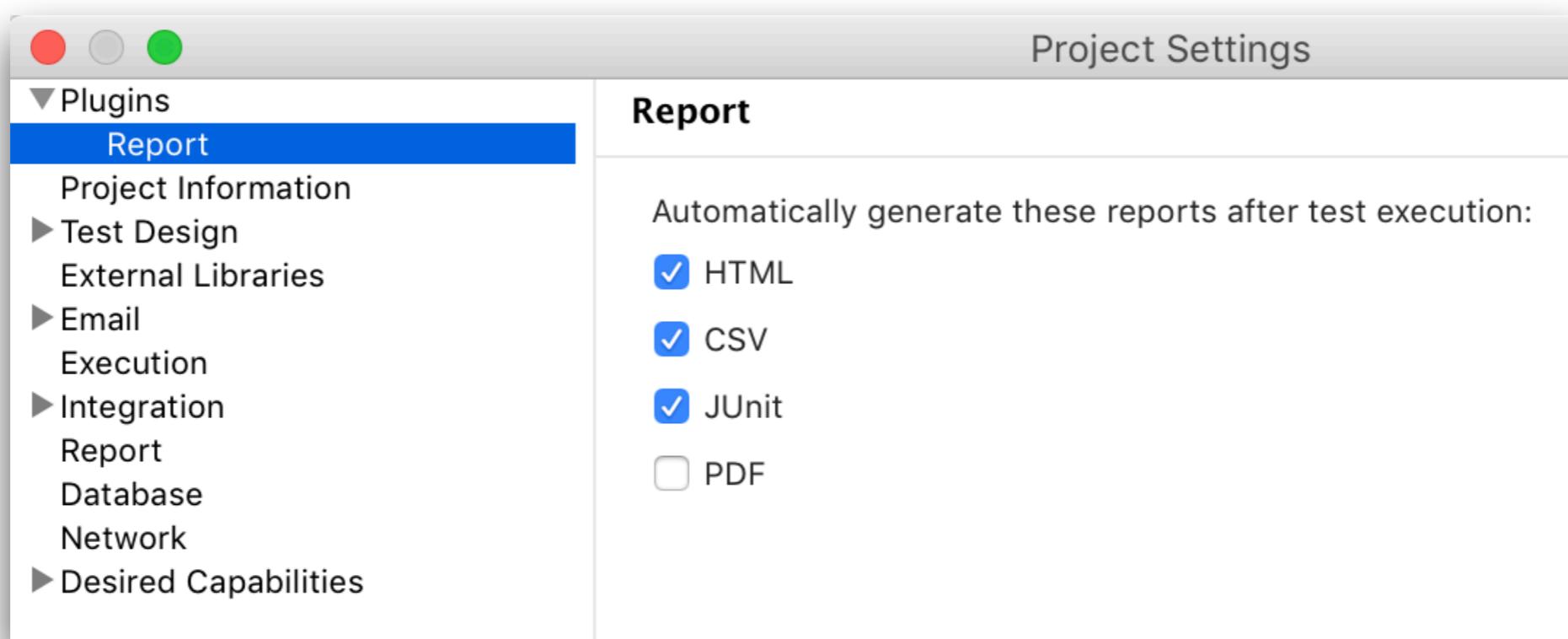


<https://store.katalon.com/product/59/Basic-Report>



# Setting report plug-in

Goto Project -> Settings

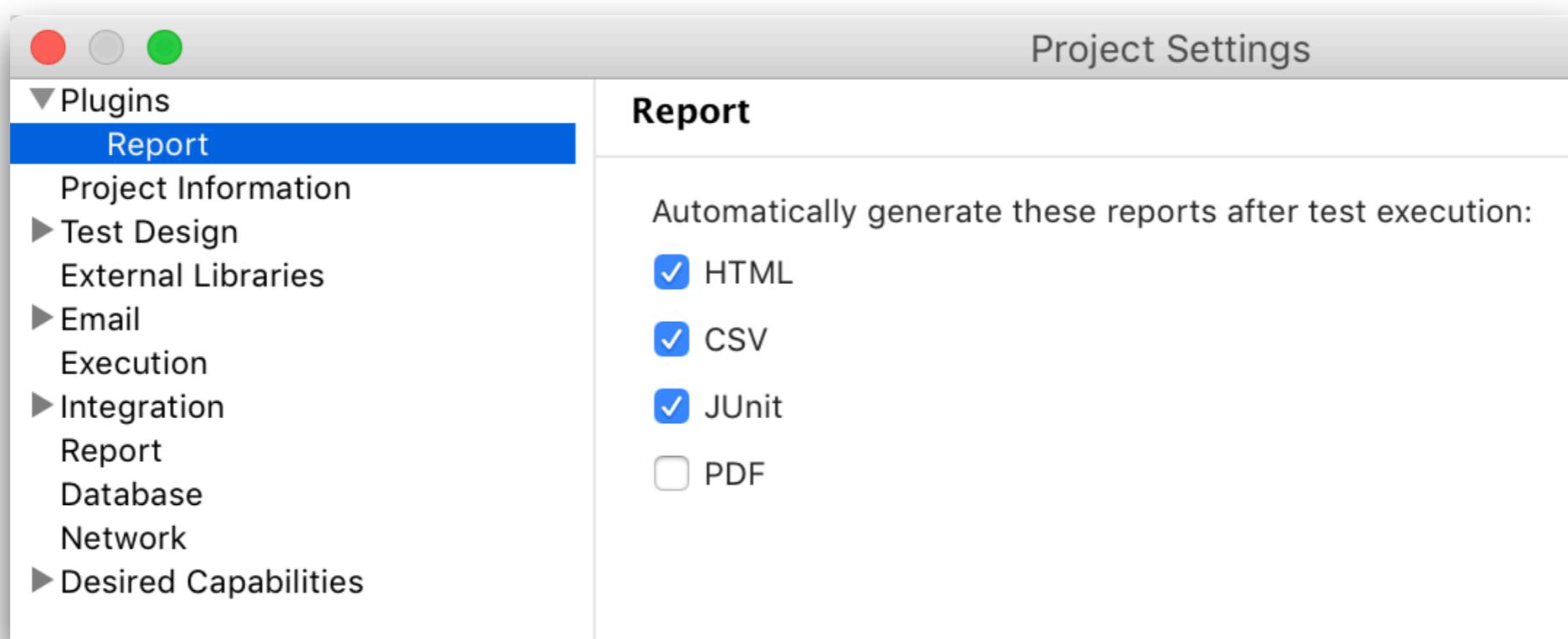


<https://docs.katalon.com/katalon-studio/docs/basic-report.html>



# Setting report plug-in

Goto Project -> Settings



<https://docs.katalon.com/katalon-studio/docs/basic-report.html>



# Run tests and see reports

## Test suite

The screenshot shows the Katalon Studio interface for a test suite named "suite\_01". At the top, there are filter buttons for Passed, Failed, Error, and Incomplete cases. Below the filters is a search bar labeled "Search here...". A table lists one test case: "Demo2 (5.912s)". To the right of the table is a "Export report" button with a dropdown menu containing four options: "HTML", "CSV (Summary)", "CSV (Details)", and "PDF". The "CSV (Details)" option is highlighted with a red box.

## Log viewer

The screenshot shows the Katalon Studio Log viewer. At the top, it displays the run statistics: Runs: 1/1, Passes: 1, Failures: 0, Errors: 0, and Skips: 0. Below this is a tree view of the log structure. The root node is "Test Suites/suite\_01 (7.153s)". Underneath it are nodes for "hostName", "os", "hostAddress", "katalonVersion", "Test Cases/Demo2 (6.817s)", and "exportKatalonReports (0.189s)". The "exportKatalonReports" node has several child entries detailing the generation of HTML, CSV, and JUnit reports.

No.	Name	Time
1	Demo2	(5.912s)

Runs: 1/1      Passes: 1      Failures: 0      Errors: 0      Skips: 0

▼ Test Suites/suite\_01 (7.153s)  
hostName = somkiat - 192.168.1.33  
os = Mac OS X 64bit  
hostAddress = 192.168.1.33  
katalonVersion = 6.3.3.11  
► Test Cases/Demo2 (6.817s)  
▼ exportKatalonReports (0.189s)  
Start generating HTML report folder at: /Users/somkiat/Katalon Studio/demo\_web/Report  
HTML report generated  
Start generating CSV report folder at: /Users/somkiat/Katalon Studio/demo\_web/Report  
CSV report generated  
Start generating JUnit report folder at: /Users/somkiat/Katalon Studio/demo\_web/Report  
JUnit report generated



# HTML report

## Execution Environment

Host name: somkiat - 192.168.1.33  
OS: Mac OS X 64bit  
Katalon version: 6.3.3.11  
Browser: Firefox 69.0

## Test Execution Log

<b>TEST SUITE:</b> suite_01	<a href="#">Expand All</a>
<b>Full Name:</b> suite_01	
<b>Start / End / Elapsed:</b> 2019-10-23 02:19:33.097 / 2019-10-23 02:19:39.130 / 00:00:06.033	
<b>Status:</b> 1 test total, 1 passed, 0 failed, 0 error, 0 incomplete	
<b>TEST CASE:</b> Test Cases/Demo2	<a href="#">Expand All</a>
<b>Full Name:</b> suite_01/Test Cases/Demo2	
<b>Start / End / Elapsed:</b> 2019-10-23 02:19:33.218 / 2019-10-23 02:19:39.130 / 00:00:05.912	
<b>Status:</b> PASSED	
<b>TEST STEP:</b> <i>openBrowser("https://www.google.com/?gws_rd=ssl")</i>	
<b>TEST STEP:</b> <i>setText(findTestObject("Page_Google/input_q"), "demo")</i>	
<b>TEST STEP:</b> <i>sendKeys(findTestObject("Page_Google/input_q"), Keys.chord(ENTER))</i>	
<b>TEST STEP:</b> <i>closeBrowser(STOP_ON_FAILURE)</i>	



# Integrate with Katalon analytics

The screenshot shows the Katalon Studio interface with the title bar "suite\_01". The main area displays the "Test Cases Table" with one entry: "Demo2 (7.461s)". Below the table are tabs for "Summary", "Execution Settings", and "Execution Environment". The "Summary" tab is selected, showing the following execution details:

Test Suite ID	Test Suites/suite_01
Host name	somkiat - 192.168.1.33
Katalon version	6.3.3.11
Start	2019-10-21 23:21:08
Elapsed	7.645s
Total TC	1
Passed	1
Error	0
Failed	0
Incomplete	0

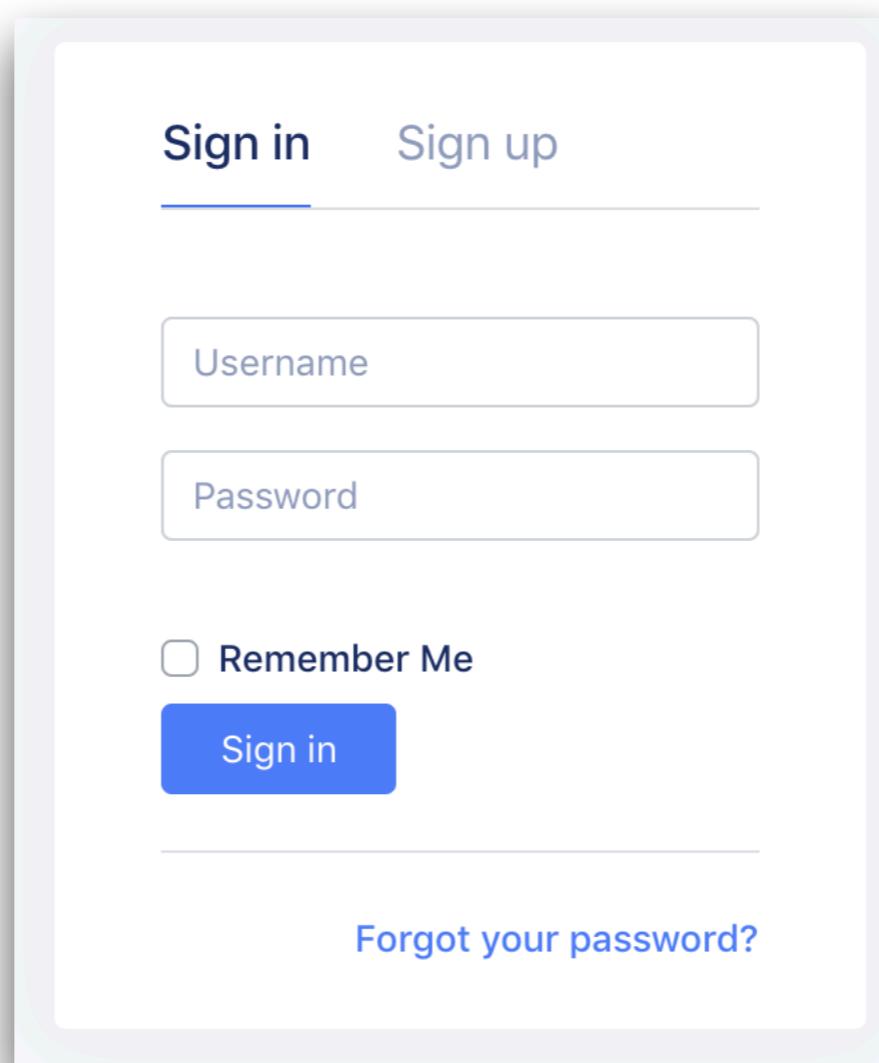
A red box highlights the "Katalon Analytics" button in the top right corner of the main window. The bottom navigation bar includes "Main", "Script", "Integration", and "Result" tabs, with "Result" being the active tab.

<https://docs.katalon.com/katalon-analytics/docs/view-reports.html>



# Katalon analytic service

Cloud service from Katalon



The image shows a screenshot of a web-based login interface. At the top, there are two buttons: "Sign in" (in blue) and "Sign up" (in grey). Below these are two input fields: "Username" and "Password". Underneath the password field is a checkbox labeled "Remember Me". A large blue button labeled "Sign in" is positioned below the "Remember Me" checkbox. At the bottom of the form, there is a link labeled "Forgot your password?".

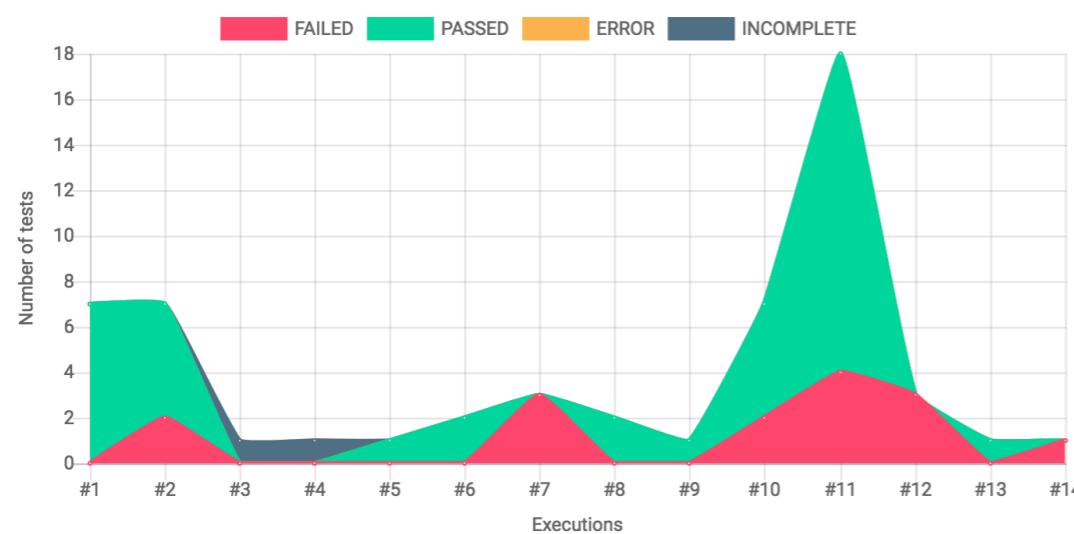
<https://analytics.katalon.com/login>



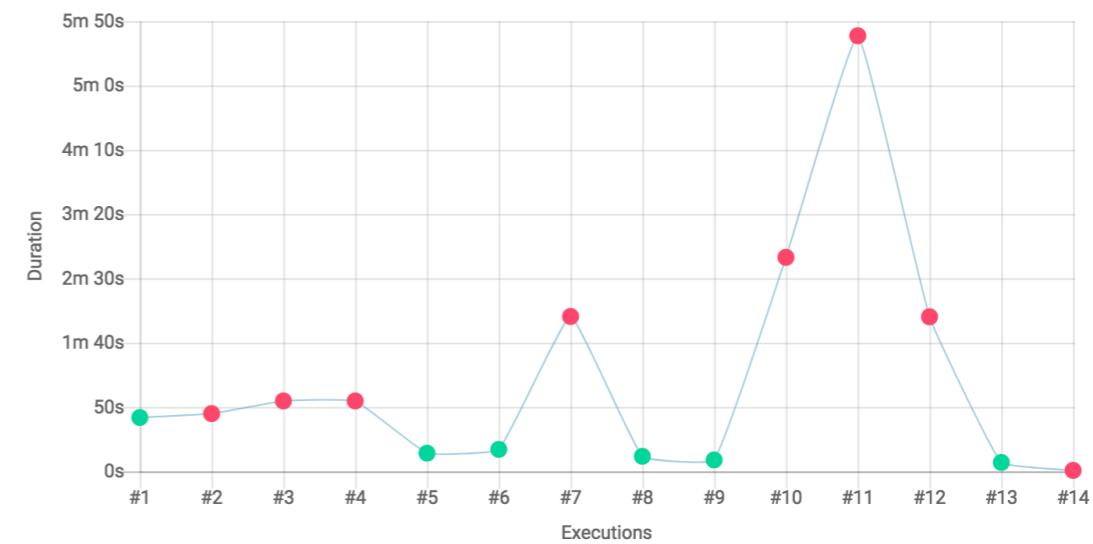
# Katalon analytic service

Project: Katalon Demo Project

## Status



## Performance



## Executions

Filter

Sort

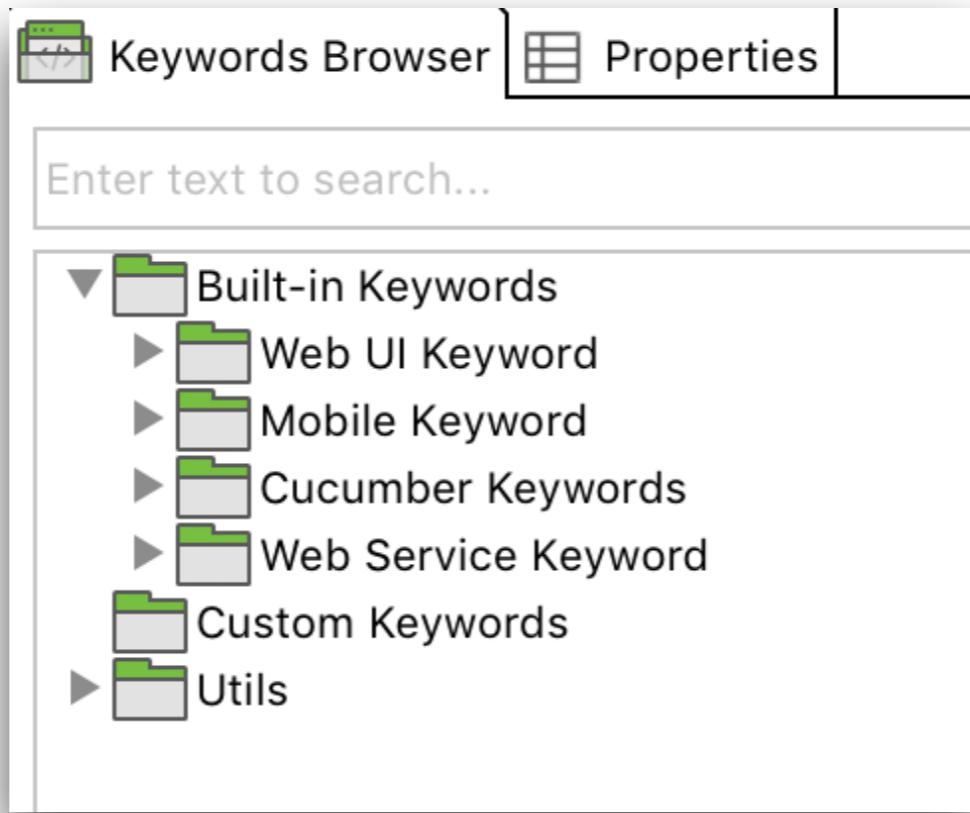
ID	Status	Test Suite	Started	Duration	Total	Passed	Failed	Error	Incomplete	Action
#14	FAILED	New Test Suite	08/01/2018 13:36:42	750ms	1	0	1	0	0	▼
#13	PASSED	Katalon Recorder Samples	03/19/2018 15:03:10	7s	1	1	0	0	0	▼
#12	FAILED	TS_Image Keywords	03/02/2018 16:31:54	2m 0s	3	0	3	0	0	▼



# Build-in keywords



# Build-in keywords



# Web UI keywords

Working with browser

Working with elements

Working with text

Upload files

Frame/Form/Checkbox/Combo box/ Alert



# Working with browser

Keyword	Description
Open browser	Open a browser and navigate to the specified URL
Navigate to Url	Navigate to the specified web page
Close browser	This action will close all windows of the browser
Maximize Window	Resize the current window to take up the entire screen
Refresh	Simulate users clicking "refresh" button on their browser

<https://docs.katalon.com/katalon-studio/docs/webui-open-browser.html>



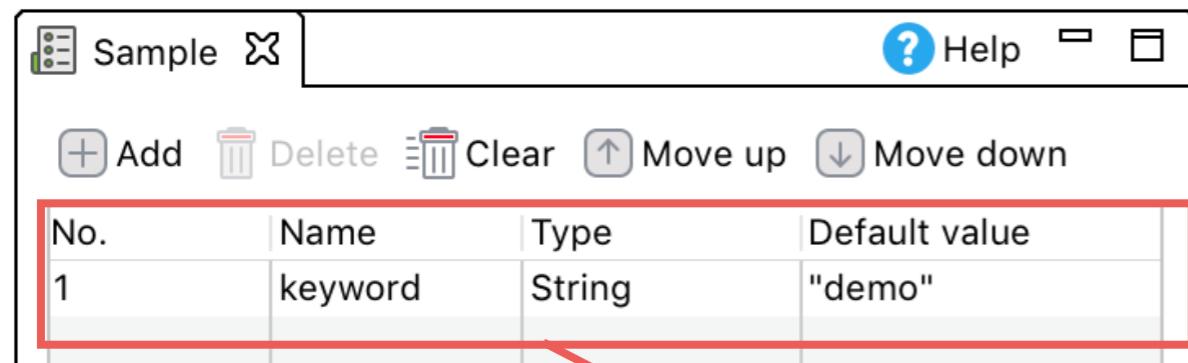
# Working with textbox

Keyword	Description
Set text	Set the value of an input field, as though you type it in
Send keys	Simulates keystroke events on the specified element
Clear text	Clear the value of an input field
Set encrypted text	Set encrypted text into an input field



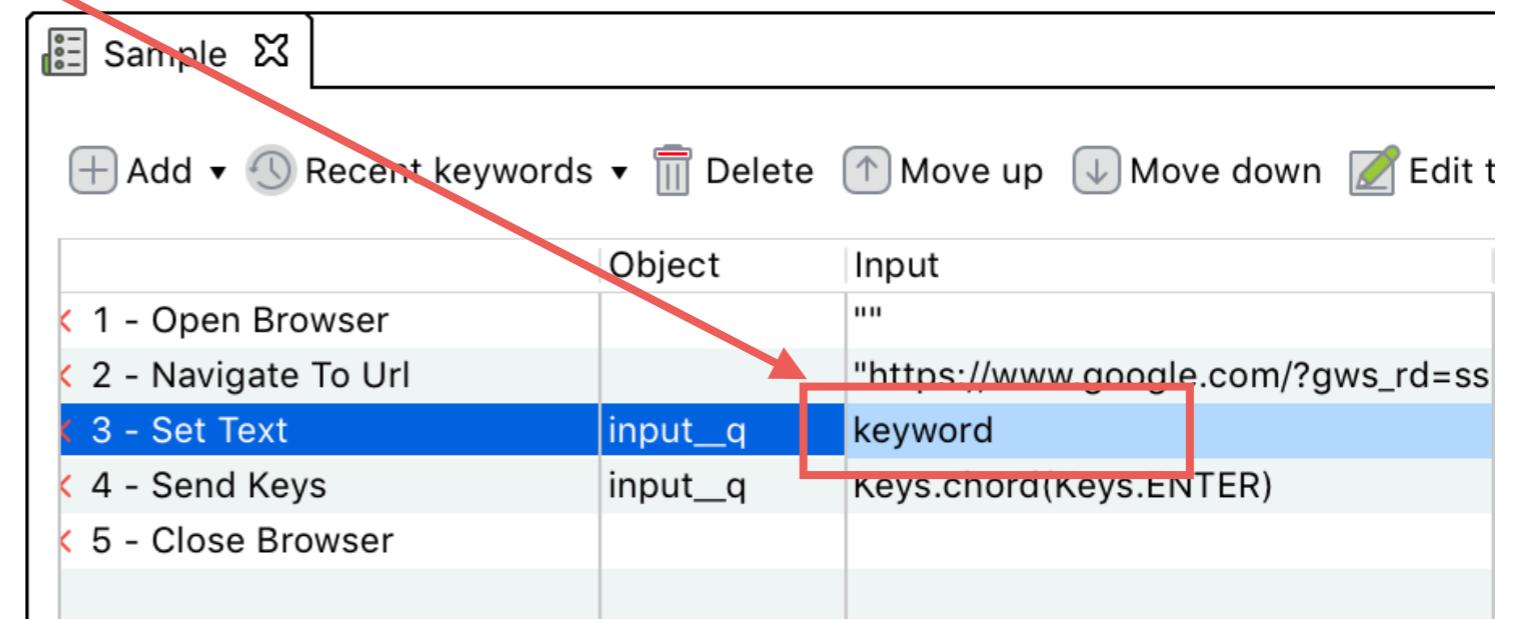
# Working with variables

## Variables tab



No.	Name	Type	Default value
1	keyword	String	"demo"

## Using in manual tab



Object	Input
1 - Open Browser	""
2 - Navigate To Url	"https://www.google.com/?gws_rd=ss-
3 - Set Text	input_q keyword
4 - Send Keys	input_q Keys.chord(Keys.ENTER)
5 - Close Browser	



# Verify elements in browser



# Verify Elements

Verify alert

Verify text

Verify element

Verify option



# Wait For

Wait for alert

Wait for element

Wait for image

Wait for page loaded



# Conditional execution of steps



# Conditional of steps

If only

If and Else

If Else-If Else

Switch-case-default



# Conditional of steps

Sample X

Add Recent keywords Delete Move up Move down Edit tags

- Web UI Keyword
- Mobile Keyword
- Cucumber Keywords
- Web Service Keyword
- Custom Keyword
- Call Test Case

Decision-making Statements ►

- If Statement
- Else Statement
- Else If Statement
- Switch Statement
- Case Statement
- Default Statement

Looping Statements

Branching Statements

Exception Handling Statements

Binary Statement

Assert Statement

Method Call Statement

Method

Sample X

Add Recent keywords Delete Move up Move down Edit tags

Item	Object	Input
IF 1 - If Statement		true
ELSE IF 2 - Else If Statement		true
ELSE 3 - Else Statement		



# Data-driven testing



# Data-driven testing

1. Create data test files
2. Define variables and add to test case
3.
  - 3.1 Create test suites
  - 3.2 Pick data from data test files
4. Execute test suites



# Data files

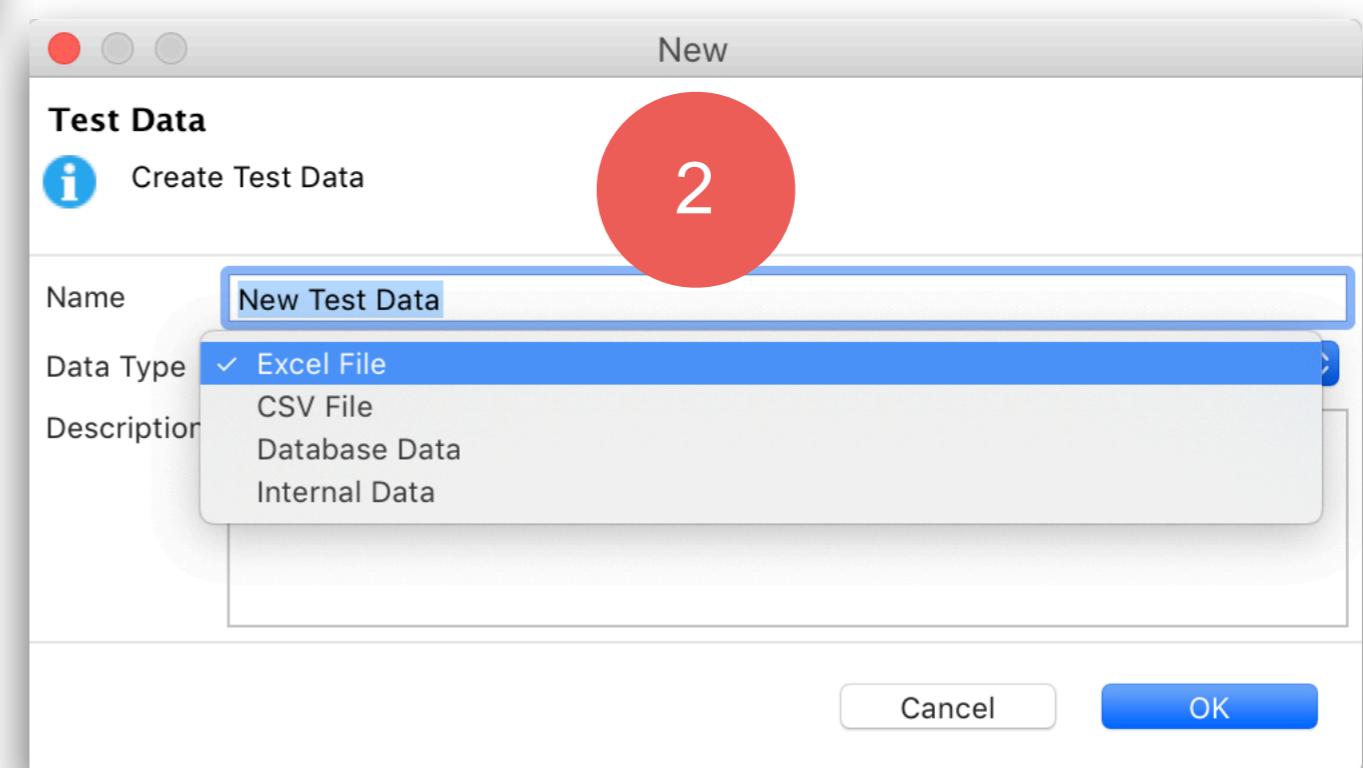
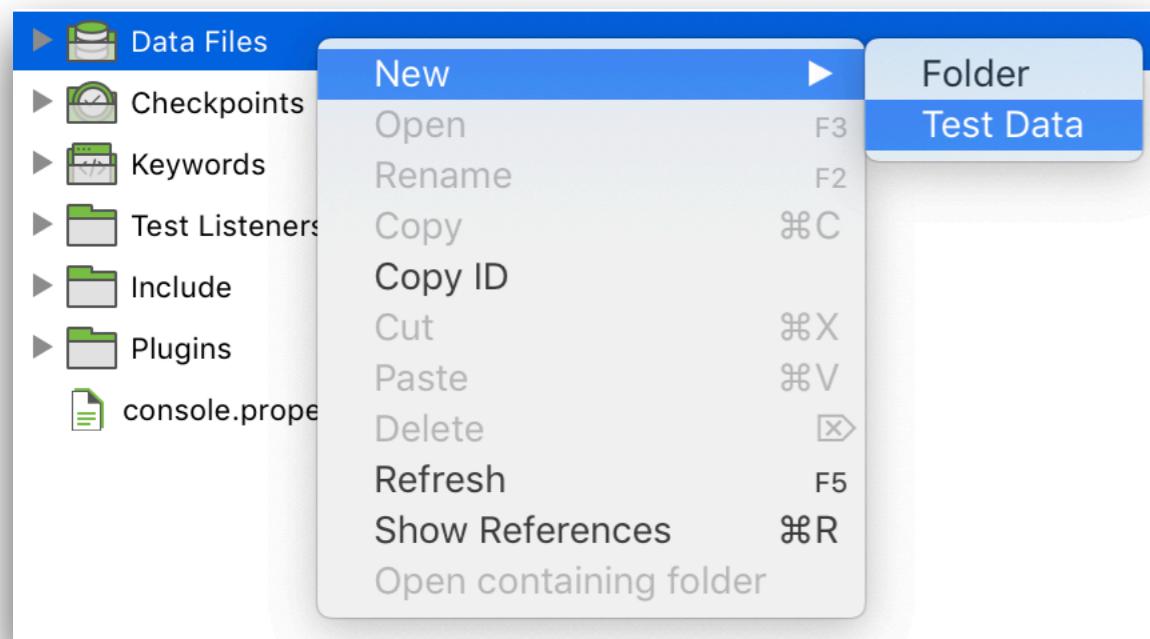
Data/ Datasource files (test data)

Different type of files

Microsoft Excel, CSV, Internal, Database



# 1. Create data files (1)



# 1. Create data files (2)

File keywords.csv

3

```
keywords.csv  
1 User,Password  
2 user01,pass  
3 user02,pass  
4 user03,pass  
5 user04,pass
```

4

Read data from CSV file

The screenshot shows a software interface for reading CSV files. At the top, there is a title bar with the text "keyword" and icons for Help, Minimize, and Close. Below the title bar, a section titled "File's Information" is expanded. It contains fields for "File Name" (set to "/Users/somkiat/data/sli"), "Separator" (set to "COMMA"), and two checkboxes: "Use first row as header" (checked) and "Use relative path" (unchecked). Below this section is a preview table showing the data from the "keywords.csv" file. The table has columns "No.", "User", and "Password". The data rows are:

No.	User	Password
1	user01	pass
2	user02	pass
3	user03	pass
4	user04	pass



# 2. Create test case and variables

## Variables tab

No.	Name	Type	Default value
1	user	String	""
2	password	String	""

**Manual tab :: using variable in input**

Item	Object	Input
1 - Open Browser		""
2 - Navigate To Url		"http://www.demo.amitjakhu.com/login"
3 - Set Text	input_Fill ou user	
4 - Set Encrypted Text	input_Fill ou password	
5 - Click	input_Fill ou	



# 3. Create test suite (1)

## Add test cases and show data binding

1 Add data file

The screenshot shows the LoginSuite application interface. On the left, there's a list of 'Execution Information' with a single entry: 'Test Cases/login'. On the right, a red box highlights the 'Test Data' section. This section contains a table with one row:

No.	ID	Data Iteration	Type
1	Data Files/keyword	All	One

Below this is a 'Variable Binding' section with two rows:

No.	Name	Default value	Type	Test Data	Value
1	user	"	Data Column	1 - Data Files/keyword	User
2	password	"	Data Column	1 - Data Files/keyword	Password

At the bottom, there are tabs for Main, Script, Integration, and Result.



# 3. Create test suite (2)

## Add test cases and show data binding

The screenshot shows the LoginSuite application interface. On the left, the 'Execution Information' panel displays a table of test cases, one of which is selected ('Test Cases/login'). On the right, the 'Test Data' panel shows a table of data files/keywords. A red box highlights the 'Variable Binding' section, which contains a table mapping variables to data columns. A large red circle with the number '2' is overlaid on the bottom right of the variable binding table.

**Execution Information**

Add Delete View Execution History Hide Data Binding

Enter text to search...			
No.	ID	Description	Run
1	Test Cases/login		<input checked="" type="checkbox"/>

**Test Data**

Add Delete Move Up Move Down Map All

No.	ID	Data Iteration	Type
1	Data Files/keyword	All	One

**Variable Binding**

Set Type Set Test Data

No.	Name	Default value	Type	Test Data	Value
1	user	"	Data Column	1 - Data Files/keyword	User
2	password	"	Data Column	1 - Data Files/keyword	Password

2 Binding data variables with test data

Main </> Script Integration Result



# 4. Execute test suite

```
Runs: 4/4      Passes: 4      Failures: 0      Errors: 0      Skips: 0      [Progress Bar]
```

▼ Test Suites/LoginSuite (8.142s)

- hostName = somkiat - 192.168.1.33
- os = Mac OS X 64bit
- hostAddress = 192.168.1.33
- katalonVersion = 6.3.3.11
- ▼ □ Test Cases/login (2.335s)
  - user = user01
  - password = pass
  - ▶ □ 1 - openBrowser("") (0.752s)
    - 2 - navigateToUrl("http://www.demo.amitjakhu.com/login-form/") (0.555s)
    - 3 - setText(findTestObject("Object Repository/Page\_Login Form/input\_Fill out the form below to login to my super awesome site"), "user01") (0.14s)
    - 4 - setText(findTestObject("Object Repository/Page\_Login Form/input\_Fill out the form below to login to my super awesome site"), "pass") (0.14s)
    - 5 - click(findTestObject("Object Repository/Page\_Login Form/input\_Fill out the form below to login to my super awesome site")) (0.055s)
  - ▶ □ Test Cases/login (1.739s)
  - ▶ □ Test Cases/login (1.741s)
  - ▶ □ Test Cases/login (1.831s)
  - ▶ □ exportKatalonReports (0.189s)



# 4. Execute test suite

## [-] TEST SUITE: LoginSuite

**Full Name:** LoginSuite  
**Start / End / Elapsed:** 2019-10-23 03:14:00.898 / 2019-10-23 03:14:09.040 / 00:00:08.142  
**Status:** 4 test total, 4 passed, 0 failed, 0 error, 0 incomplete

## [-] TEST CASE: Test Cases/login

**Full Name:** LoginSuite/Test Cases/login  
**Start / End / Elapsed:** 2019-10-23 03:14:01.079 / 2019-10-23 03:14:03.414 / 00:00:02.335  
**Status:** PASSED

[+] TEST STEP: `openBrowser("")`  
[+] TEST STEP: `navigateToUrl("http://www.demo.amitjakhu.com/login-form/")`  
[+] TEST STEP: `setText(findTestObject("Object Repository/Page_Login Form/input_Fill out the form below to login to my super awesome imaginary control panel_username"), user)`  
[+] TEST STEP: `setText(findTestObject("Object Repository/Page_Login Form/input_Fill out the form below to login to my super awesome imaginary control panel_password"), password)`  
[+] TEST STEP: `click(findTestObject("Object Repository/Page_Login Form/input_Fill out the form below to login to my super awesome imaginary control panel_submit"))`

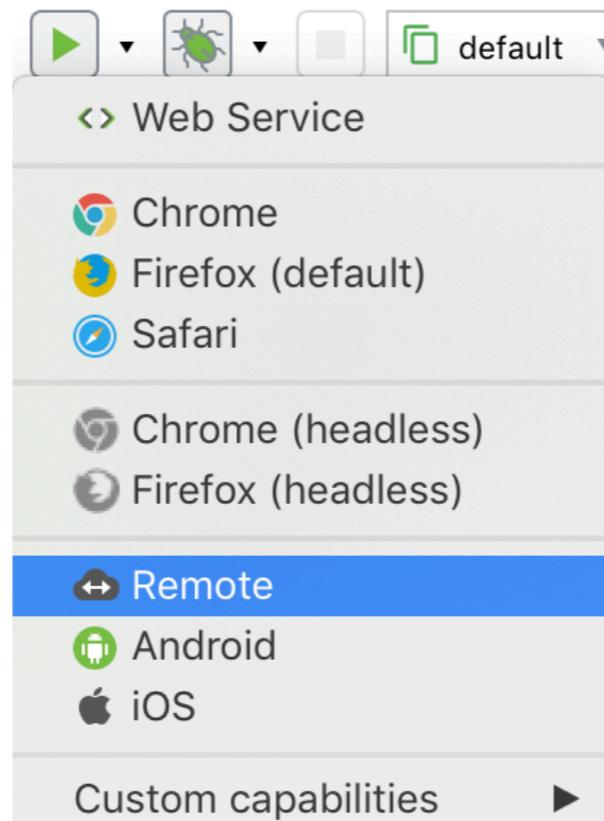
## [-] TEST CASE: Test Cases/login

**Full Name:** LoginSuite/Test Cases/login  
**Start / End / Elapsed:** 2019-10-23 03:14:03.430 / 2019-10-23 03:14:05.169 / 00:00:01.739  
**Status:** PASSED

[+] TEST STEP: `openBrowser("")`  
[+] TEST STEP: `navigateToUrl("http://www.demo.amitjakhu.com/login-form/")`  
[+] TEST STEP: `setText(findTestObject("Object Repository/Page_Login Form/input_Fill out the form below to login to my super awesome imaginary control panel_username"), user)`  
[+] TEST STEP: `setText(findTestObject("Object Repository/Page_Login Form/input_Fill out the form below to login to my super awesome imaginary control panel_password"), password)`  
[+] TEST STEP: `click(findTestObject("Object Repository/Page_Login Form/input_Fill out the form below to login to my super awesome imaginary control panel_submit"))`

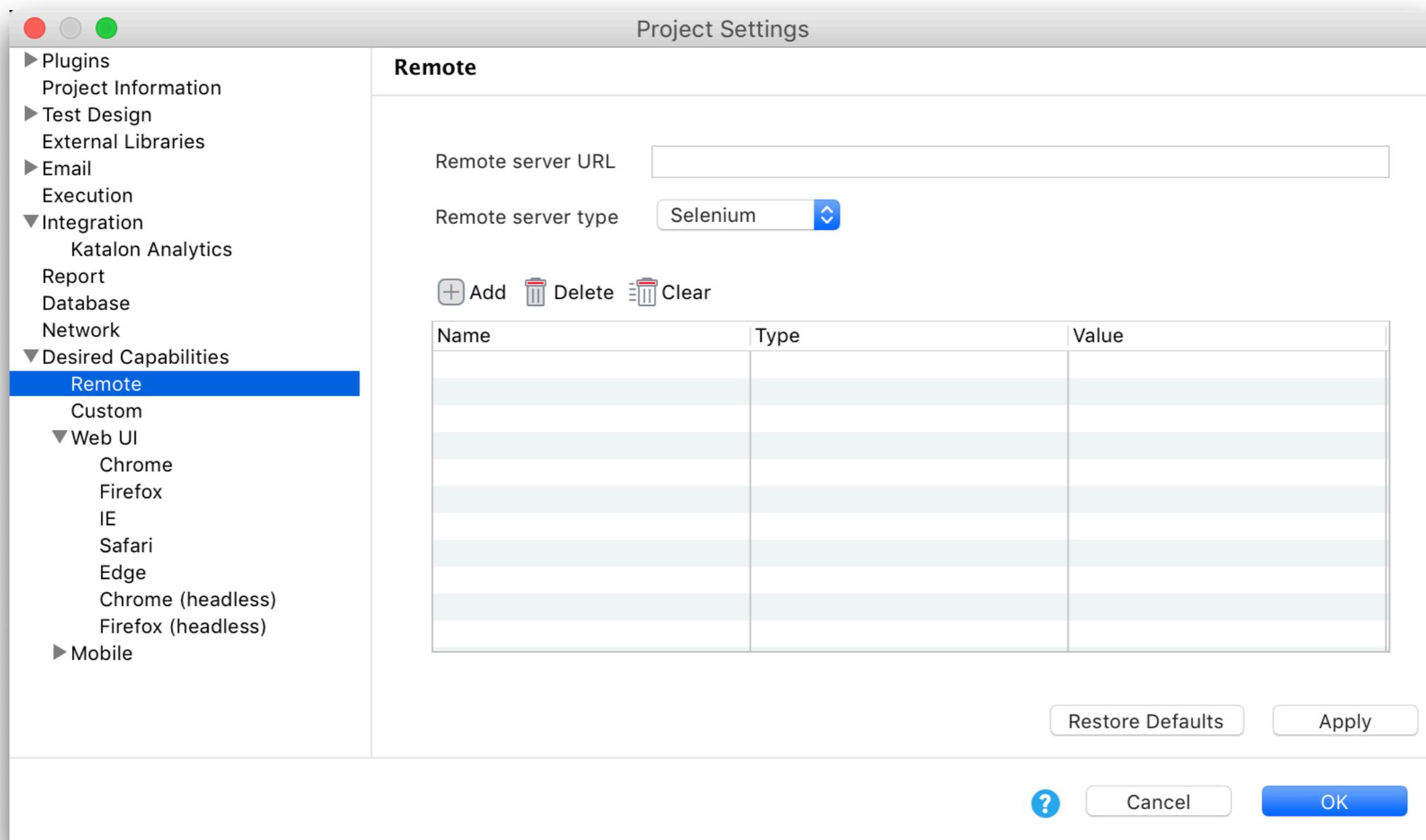


# Katalon with Remote

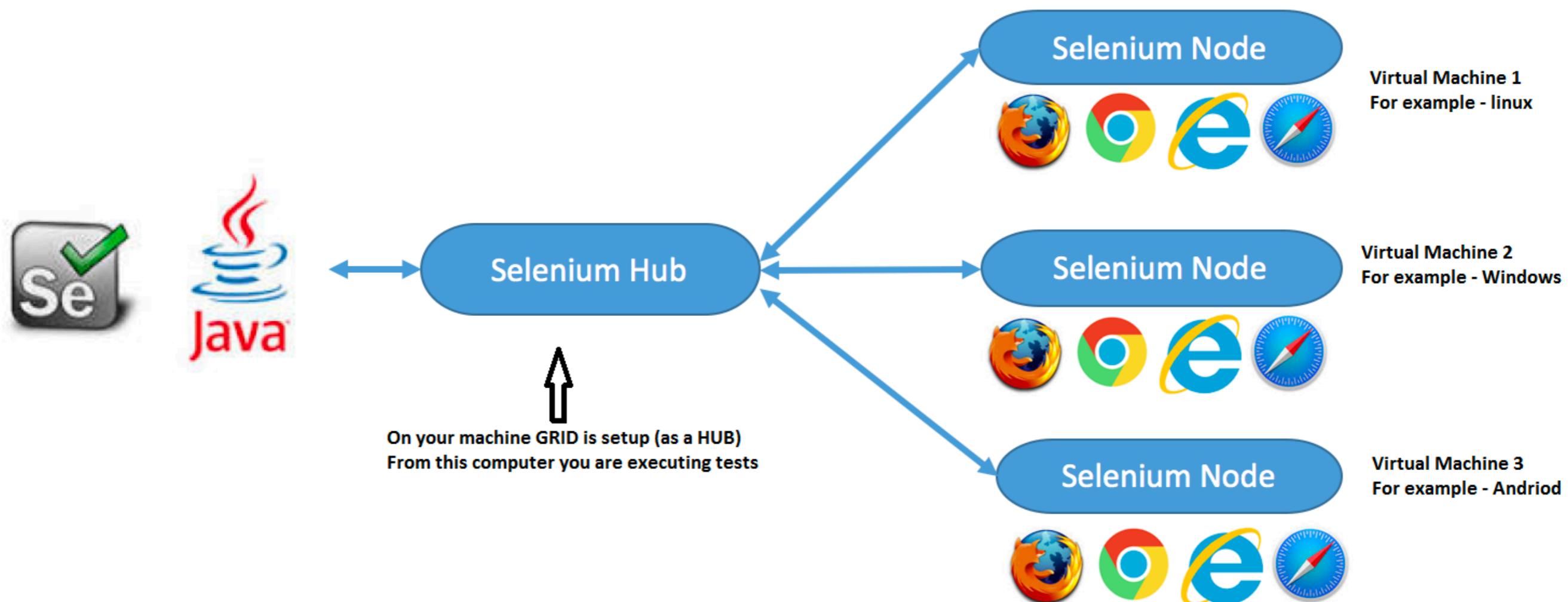


# Project setting

## Add remote url of Selenium Grid



# Selenium Grid



<https://github.com/SeleniumHQ/selenium/wiki/Grid2>



# API Testing



# Import from other system

Import from swagger

Import from Postman

Import from WSDL (Web Service)



# API Testing

Create REST request  
Validate status code  
Validate response data



# 1. Create REST request

Save new request to Object repository

The screenshot shows the Postman application interface. On the left, the request details are set to a GET method at <https://jsonplaceholder.typicode.com/users>. The 'Query Parameters' section is expanded, showing an empty table for adding parameters. Below the table are tabs for Authorization, HTTP Header, HTTP Body, Verification, Variables, Variables Editor, and Configuration. Under Authorization, it says 'No Authorization'. Under Configuration, there is a dropdown menu with 'Update to HTTP Header'.

On the right, the 'Response' panel displays the API's response. The status is 200 OK, elapsed time is 207 ms, and size is 6 KB. The response body is shown in JSON format, listing one user object:

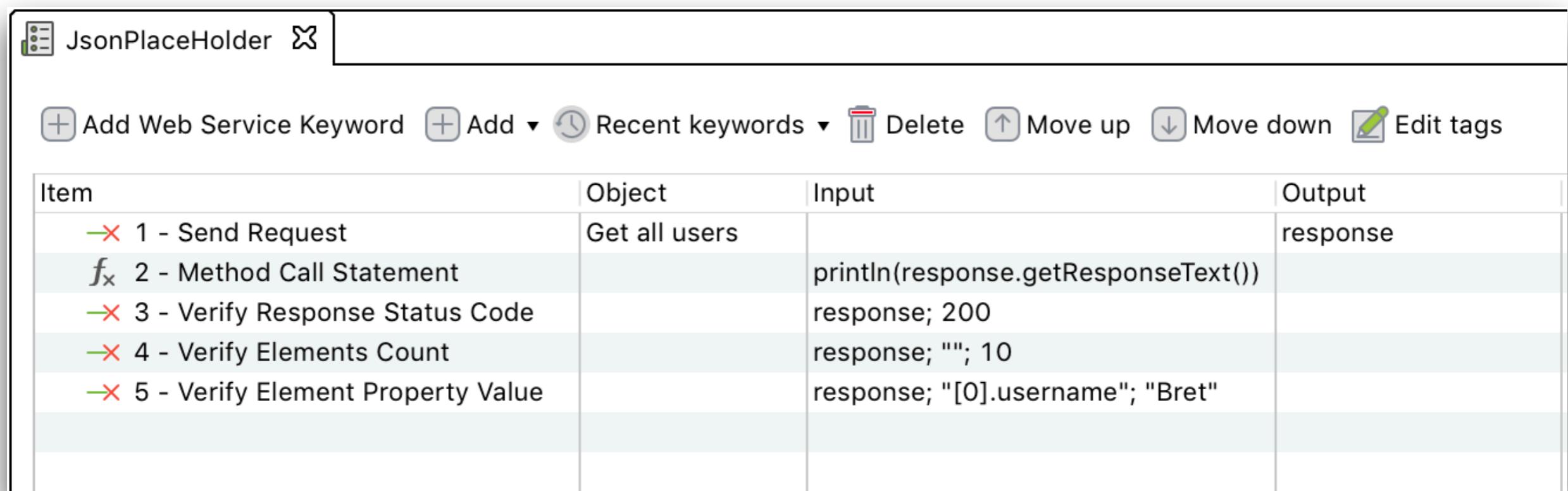
```
[  
  {  
    "id": 1,  
    "name": "Leanne Graham",  
    "username": "Bret",  
    "email": "Sincere@april.biz",  
    "address": {  
      "street": "Kulas Light",  
      "suite": "Apt. 556",  
      "city": "Gwenborough",  
      "zipcode": "92998-3874",  
      "geo": {  
        "lat": "-37.3159",  
        "lng": "81.1496"  
      }  
    }  
]
```

Below the JSON, there is a note: 'Select JSON or XML response data and press Ctrl/Command + K to add verification scripts directly.' At the bottom of the response panel, there are radio buttons for JSON (selected), XML, HTML, JavaScript, and a checked checkbox for 'Wrap Line'.



# 2. Create new test case

Save new request to Object repository  
Add Web Service Keyword



Item	Object	Input	Output
1 - Send Request	Get all users		response
2 - Method Call Statement		println(response.getText())	
3 - Verify Response Status Code		response; 200	
4 - Verify Elements Count		response; ""; 10	
5 - Verify Element Property Value		response; "[0].username"; "Bret"	



# 3. Validate result

The screenshot shows a test tool interface with a toolbar at the top and a table below it. The toolbar includes icons for adding a web service keyword, adding recent keywords, deleting, moving up, moving down, and editing tags. The table has four columns: Item, Object, Input, and Output. The rows represent the following steps:

Item	Object	Input	Output
1 - Send Request	Get all users		response
2 - Method Call Statement		println(response.getText())	
3 - Verify Response Status Code		response; 200	
4 - Verify Elements Count		response; ""; 10	
5 - Verify Element Property Value		response; "[0].username"; "Bret"	



# Print data to console

```
+ import static com.kms.katalon.core.checkpoint.CheckpointFactory.findCheckpoint()
+ response = WS.sendRequest(findTestObject('Get all users'))
println(response.getText())
WS.verifyResponseStatusCode(response, 200)
WS.verifyElementsCount(response, '', 10)
WS.verifyElementPropertyValue(response, '[0].username', 'Bret')
```



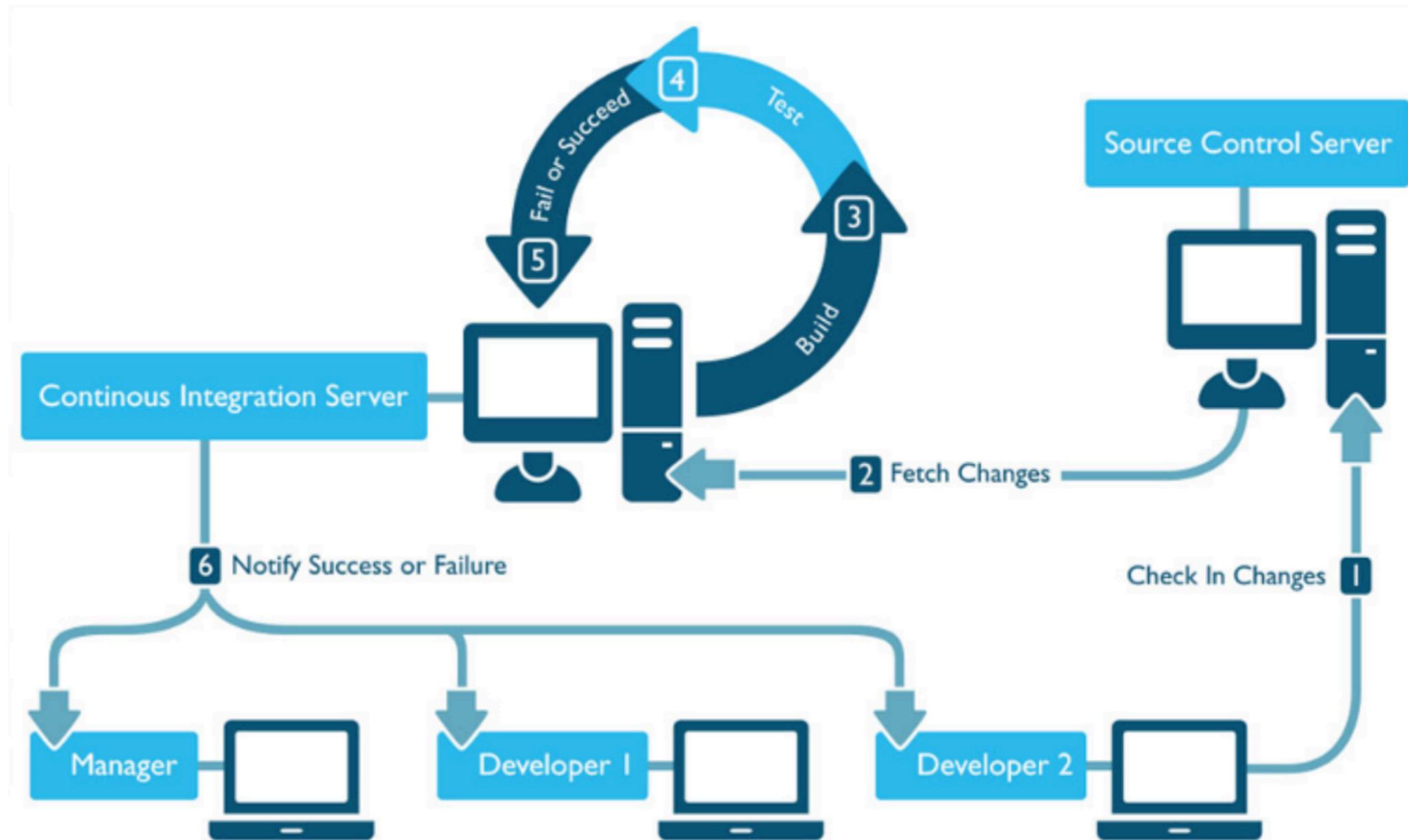
# Continuous Testing



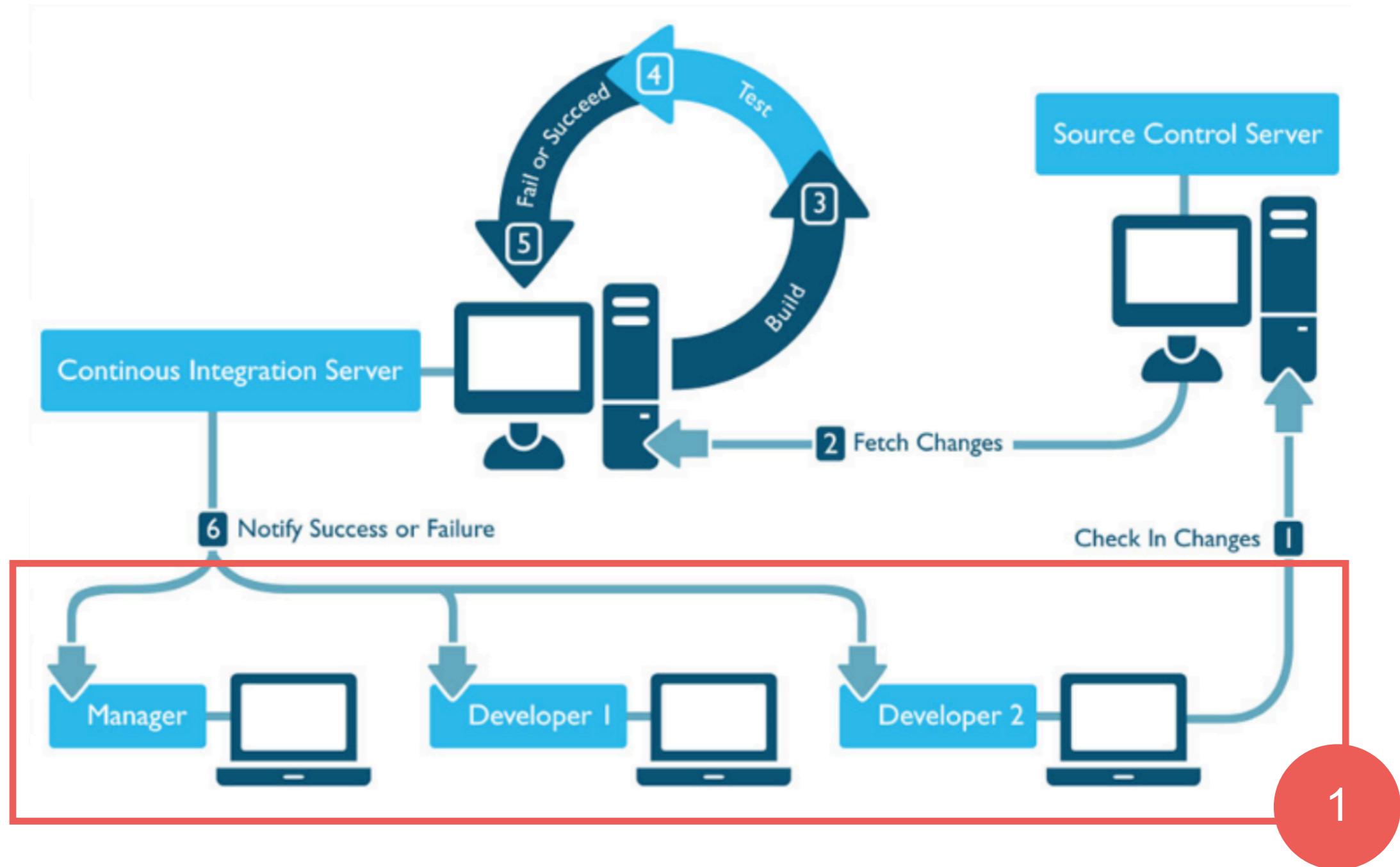
<https://jenkins.io/>



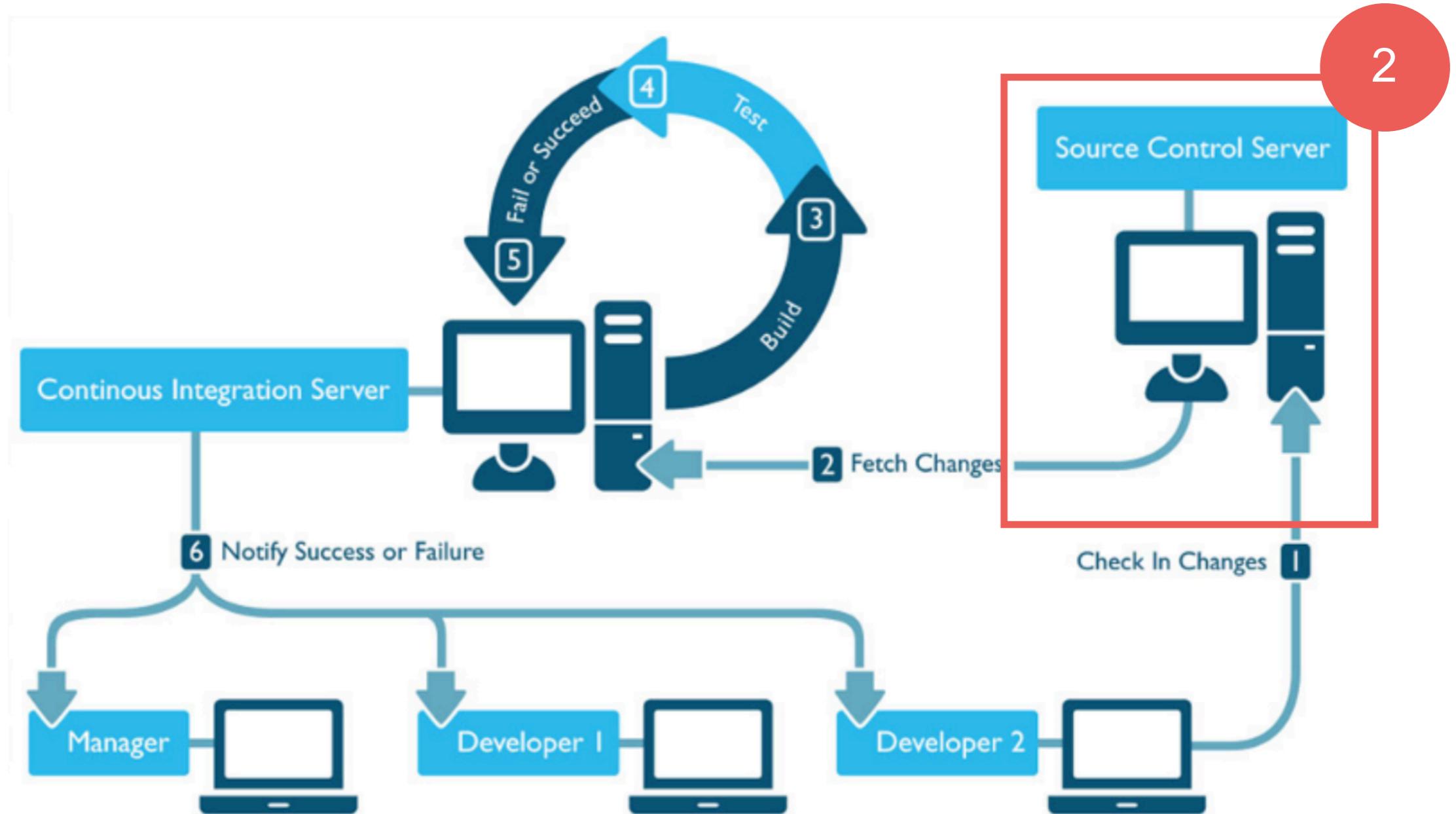
# Continuous Integration and Testing



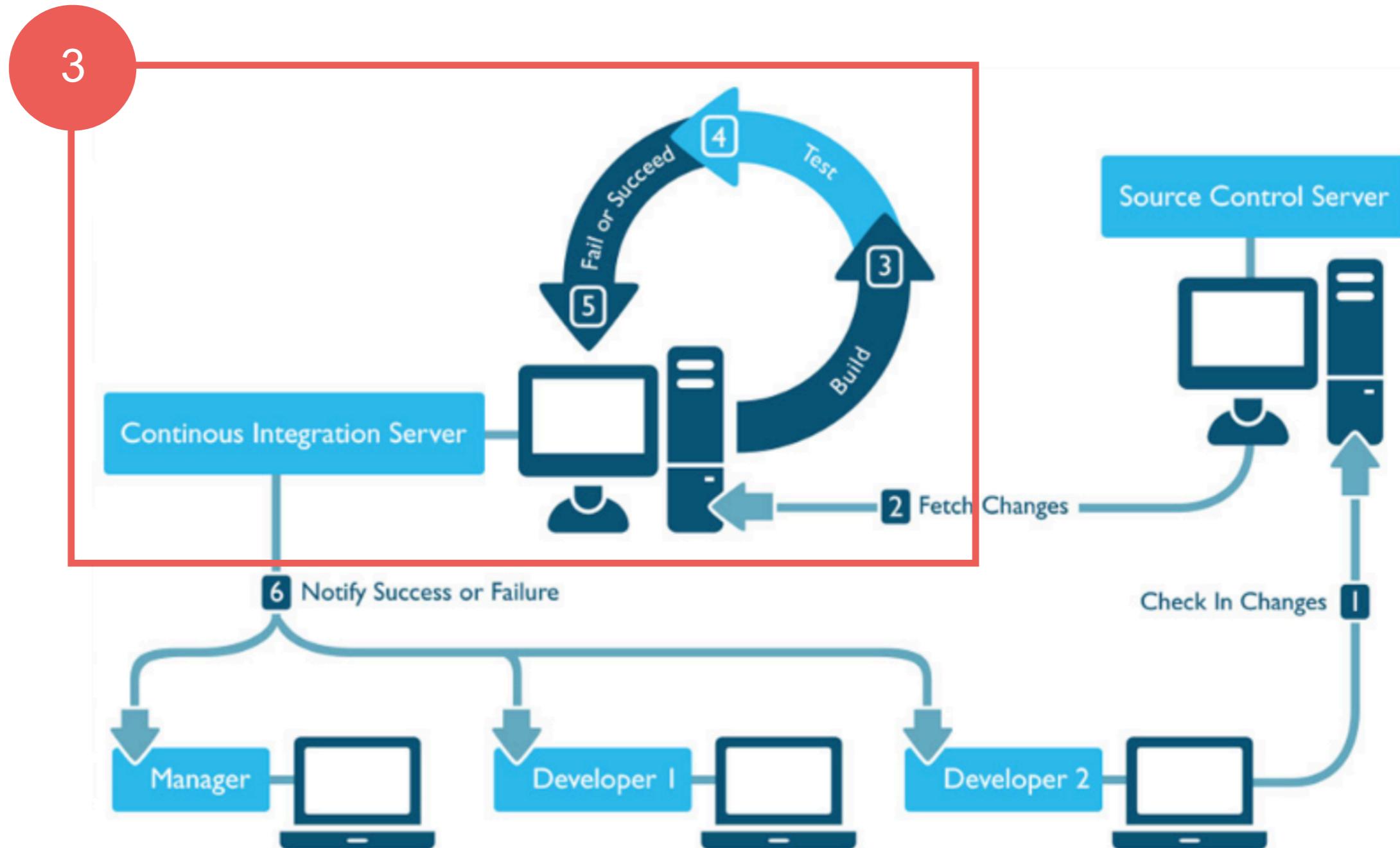
# Continuous Integration and Testing



# Continuous Integration and Testing



# Continuous Integration and Testing



# Start Jenkins server

```
$java -jar jenkins.war
```

Jenkins initial setup is required. An admin user  
been created and a password generated.  
Please use the following password to proceed to :  
llation:

```
07be76a843f54c3bae1d259ee38ed2d9
```

This may also be found at: /Users/somkiat/data/s  
robot-framework/thaibev-20180806/ci/v2/secrets/i  
lAdminPassword



# Open in browser

http://localhost:8080

The screenshot shows the Jenkins 'Getting Started' screen. The main title is 'Unlock Jenkins'. A text block explains that a password has been written to the log and on the server. It provides a redacted log path: '/Users/somkiat/data/slide/robot-framework/thaibev-20180806/ci/v2/secrets/initialAdminPassword'. Below this, a placeholder for the 'Administrator password' is shown with a redacted input field. A 'Continue' button is at the bottom right.

Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

```
/Users/somkiat/data/slide/robot-framework/thaibev-  
20180806/ci/v2/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue



# Install suggested plugins

Getting Started X

## Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

**Install suggested  
plugins**

Install plugins the Jenkins community finds most useful.

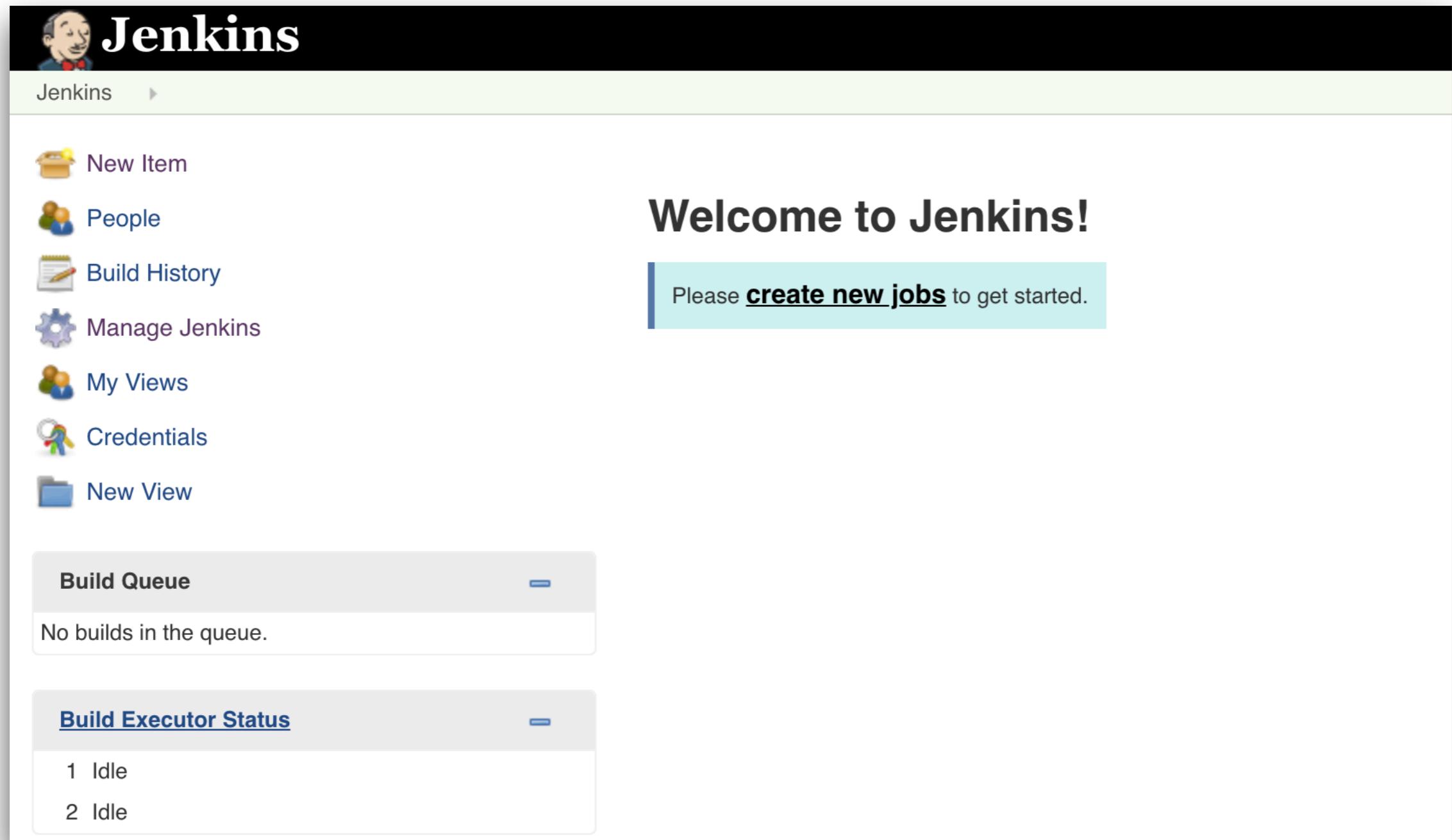
**Select plugins to  
install**

Select and install plugins most suitable for your needs.

Jenkins 2.121.3



# Ready to start !!



The image shows the Jenkins web interface. At the top left is the Jenkins logo. The main header says "Welcome to Jenkins!". Below it, a message says "Please [create new jobs](#) to get started." On the left sidebar, there are several links: "New Item", "People", "Build History", "Manage Jenkins", "My Views", "Credentials", and "New View". Under "Build Queue", it says "No builds in the queue." Under "Build Executor Status", it lists "1 Idle" and "2 Idle".



# 1. Create new job

Enter an item name  
**Fill in job name**  
» Required field

 **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

 **Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**OK**



# 2. Source code management

General    **Source Code Management**    Build Triggers    Build Environment    Build    Post-build Actions

## Source Code Management

None  
 Git  
 Subversion

(?)

## Build Triggers

Trigger builds remotely (e.g., from scripts) (?)  
 Build after other projects are built (?)  
 Build periodically (?)  
 GitHub hook trigger for GITScm polling (?)  
 Poll SCM (?)



# 3. Build trigger

General    **Source Code Management**    Build Triggers    Build Environment    Build    Post-build Actions

---

## Source Code Management

None  
 Git  
 Subversion ?

---

## Build Triggers

Trigger builds remotely (e.g., from scripts) ?  
 Build after other projects are built ?  
 Build periodically ?  
 GitHub hook trigger for GITScm polling ?  
 Poll SCM ?



# 4. Build trigger with poll SCM

Poll SCM  

Schedule

\* \* \* \* \*

No schedules so will only run due to SCM changes if triggered by a post-commit hook

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0–59)

HOUR The hour of the day (0–23)

DOM The day of the month (1–31)

MONTH The month (1–12)

DOW The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of precedence,

- \* specifies all valid values
- M–N specifies a range of values
- M–N/X or \*/X steps by intervals of X through the specified range or whole valid range
- A, B, . . . , Z enumerates multiple values



# 5. Build your job

Build

Add build step ▾

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit



# 6. Post build actions

**Post-build Actions**

Add post-build action ▾

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Publish Robot Framework test results**
- Record fingerprints of files to track usage
- Git Publisher
- E-mail Notification
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done



# 7. Add post build with Robot framework

**Post-build Actions**

**Publish Robot Framework test results**

Directory of Robot output

Path to directory containing robot xml and html files (relative to build workspace)

Advanced...

Thresholds for build result

Yellow %  **Entry must be percentage value between 0-100**

Blue %  **Entry must be percentage value between 0-100**

Use thresholds for critical tests only

Add post-build action ▾



# 8. Install Katalon plugin

The screenshot shows the Jenkins 'Manage Jenkins' interface. On the left, there's a sidebar with links: 'People', 'Build History', 'Manage Jenkins' (which is highlighted with a red box and has a red circle with '1' above it), 'My Views', 'Credentials', and 'New View'. Below that are two sections: 'Build Queue' (empty) and 'Build Executor Status' (showing 1 Idle and 2 Idle executors). The main content area is titled 'Manage Jenkins' and contains several configuration options, each with an icon and a brief description. A red box highlights the 'Manage Plugins' option, which has a red circle with '2' above it and is also enclosed in a red rectangle. The other options are: 'Configure System' (gear icon), 'Configure Global Security' (padlock icon), 'Configure Credentials' (key icon), 'Global Tool Configuration' (wrench and screwdriver icon), 'Reload Configuration from Disk' (refresh icon), 'System Information' (monitor icon), and 'System Log' (clipboard icon).

**Manage Jenkins**

- Configure System**  
Configure global settings and paths.
- Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.
- Configure Credentials**  
Configure the credential providers and types
- Global Tool Configuration**  
Configure tools, their locations and automatic installers.
- Reload Configuration from Disk**  
Discard all the loaded data in memory and reload everything from file system. Useful when you modify disk.
- Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- System Information**  
Displays various environmental information to assist trouble-shooting.
- System Log**  
System log captures output from `java.util.logging` output related to Jenkins.



# 8. Install Katalon plugin

The screenshot shows the Jenkins plugin manager interface. At the top, there are tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A search bar labeled 'Filter:' contains the text 'katalon'. Below the tabs, a table lists a single plugin:

Install ↓	Name	Version
<input type="checkbox"/> <a href="#">Katalon Studio</a> Execute Katalon Studio in Jenkins		1.0.18

At the bottom of the screen, there are three buttons: 'Install without restart', 'Download now and install after restart' (which is highlighted in blue), and 'Check now'. A status message says 'Update information obtained: 21 sec ago'.

<https://plugins.jenkins.io/katalon>



# Robot Framework



<https://robotframework.org/>



# Official website



INTRODUCTION  
EXAMPLES  
LIBRARIES  
TOOLS  
DOCUMENTATION  
SUPPORT  
FOUNDATION  
SHOP  
ROBOCON

# ROBOT FRAME WORK /

## INTRODUCTION

**Robot Framework** is a generic test automation framework for acceptance testing and acceptance test-driven development (ATDD). It has easy-to-use tabular test data



# Robot Framework

Test automation framework

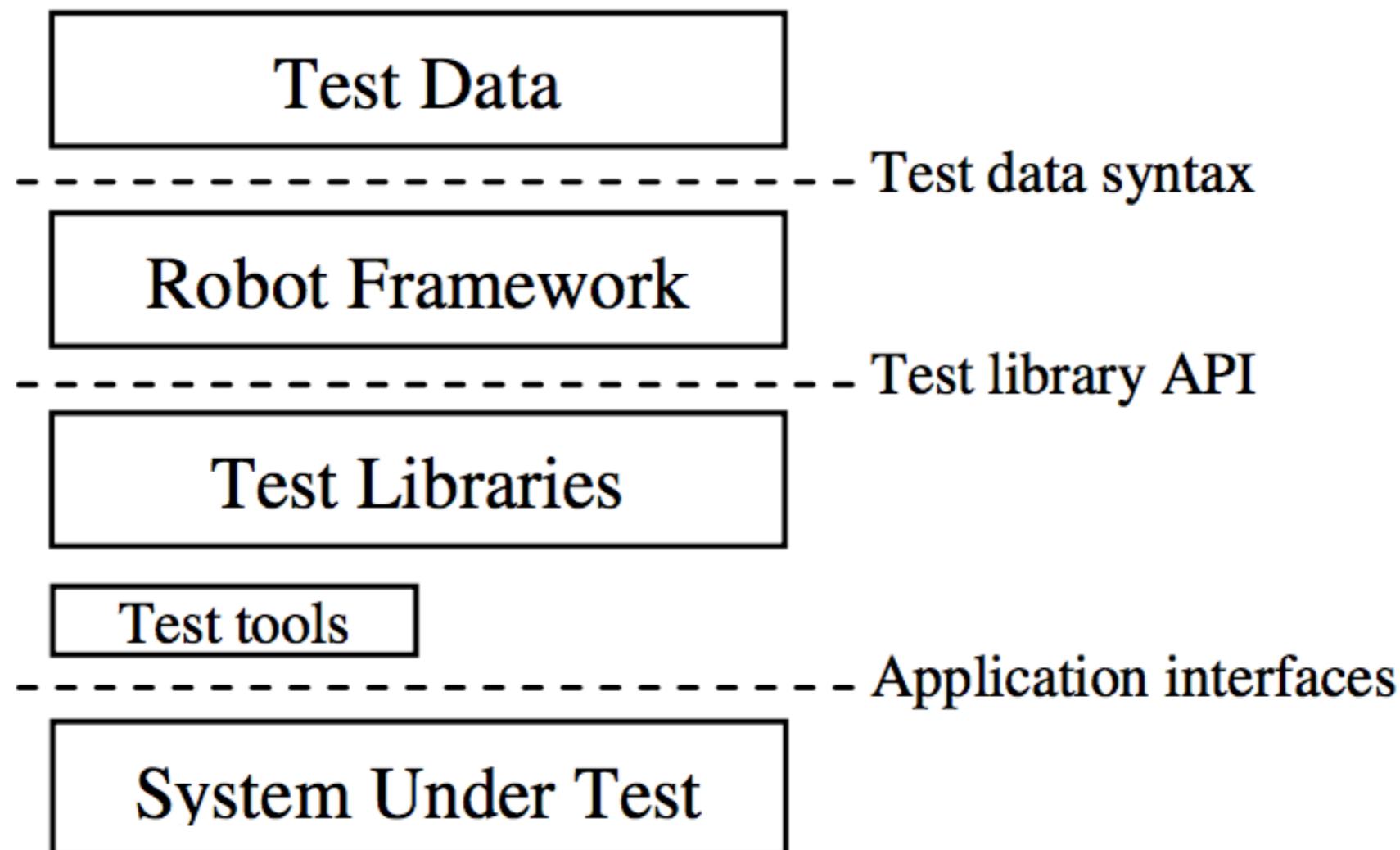
Designed for Acceptance testing(ATDD)

Developed with Python

Compatible with Java (Jython)



# Robot Framework architecture



# Robot Framework architecture (1)

Robot Framework Core  
(standard libraries, reporting)

External library

External library

Resource  
file 1

Resource  
file 2

Resource  
file n

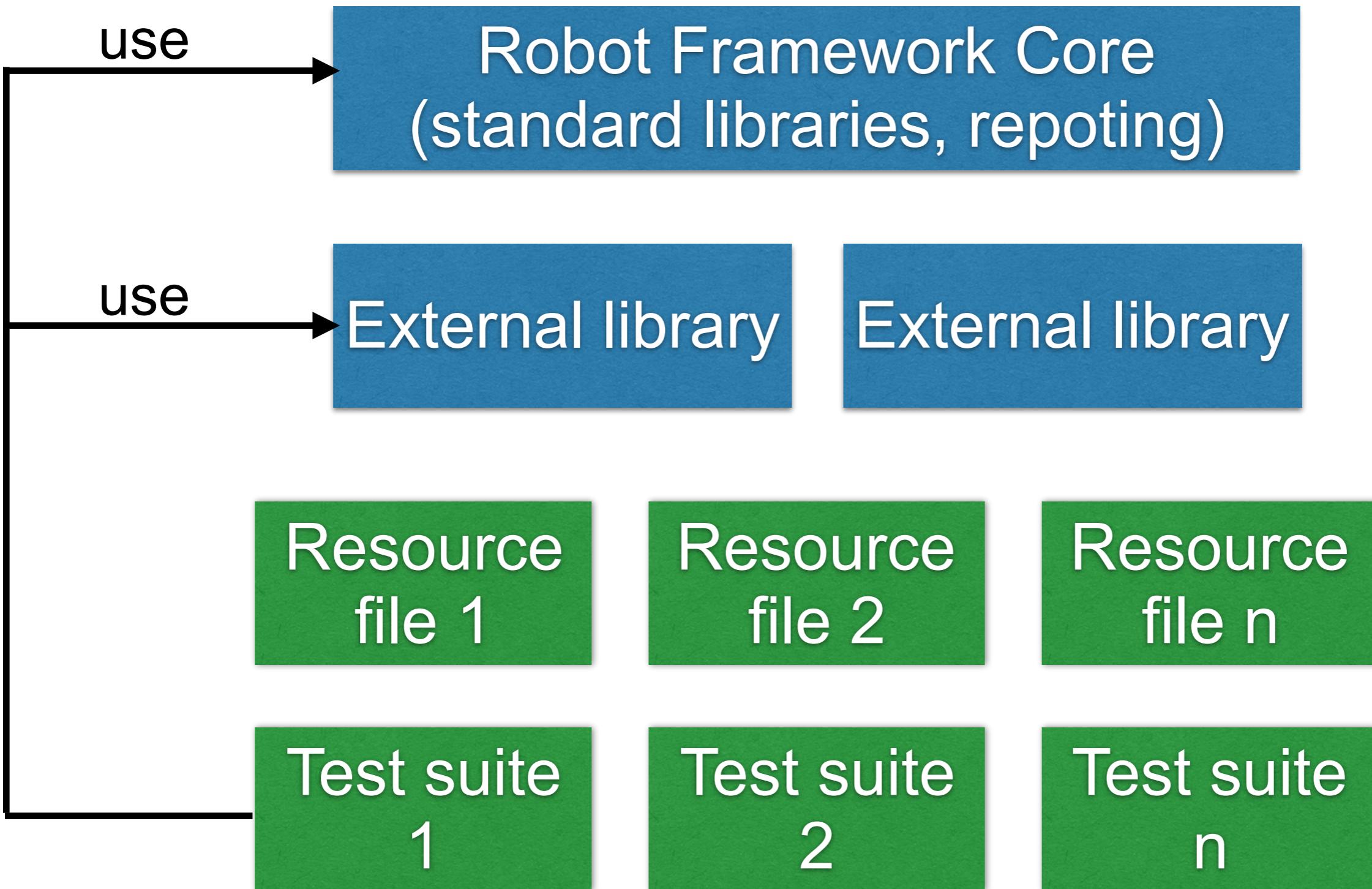
Test suite  
1

Test suite  
2

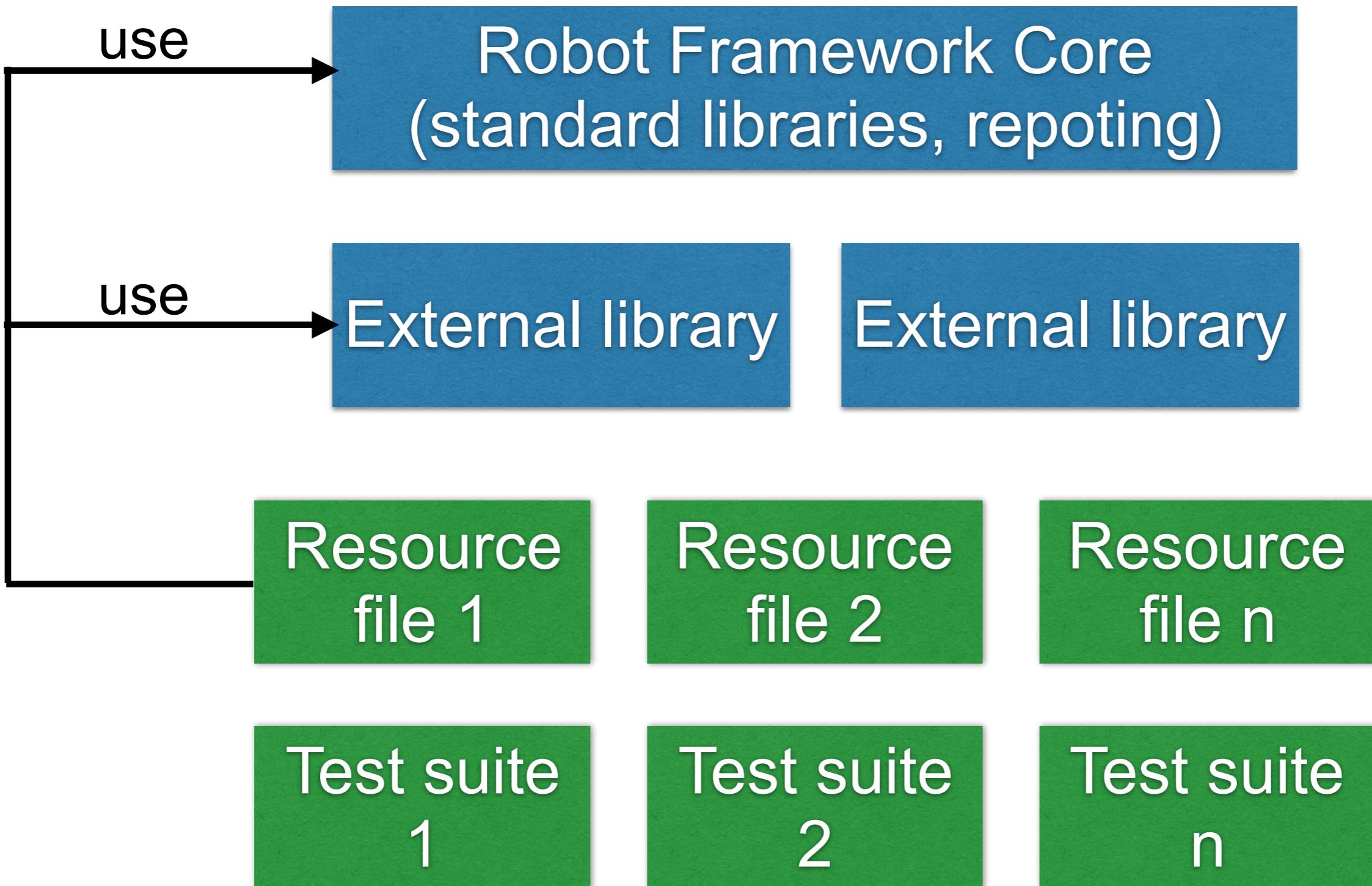
Test suite  
n



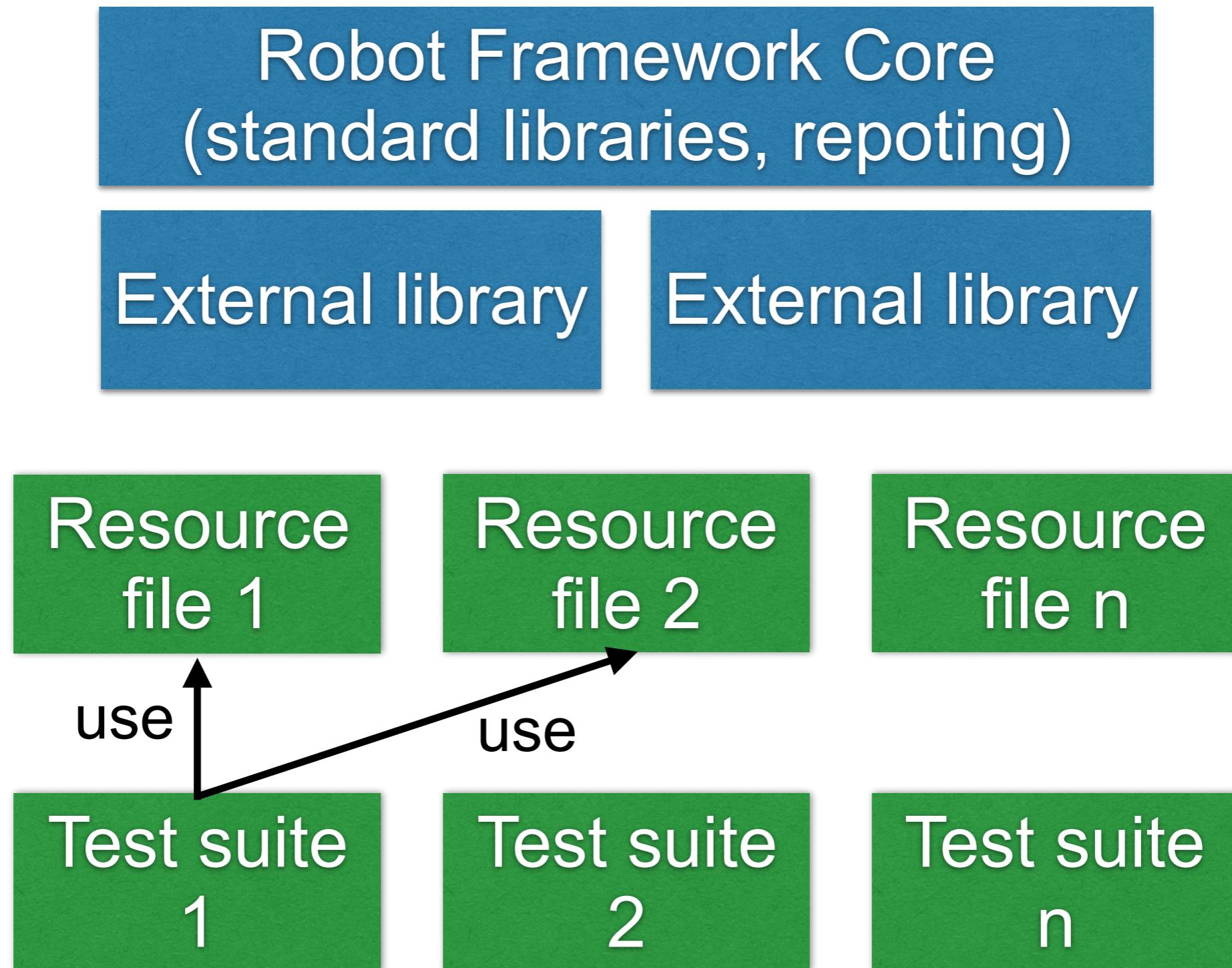
# Robot Framework architecture (2)



# Robot Framework architecture (3)



# Robot Framework architecture (4)



# **Recommended Libraries**

## **Selenium Library**

Web testing library that uses popular Selenium tool internally

## **HTTP Library (requests)**

Library for HTTP level testing using Request internally

## **REST Instance**

Robot Framework test library for HTTP JSON APIs

## **Faker Library**

Library for Faker, a fake test data generator



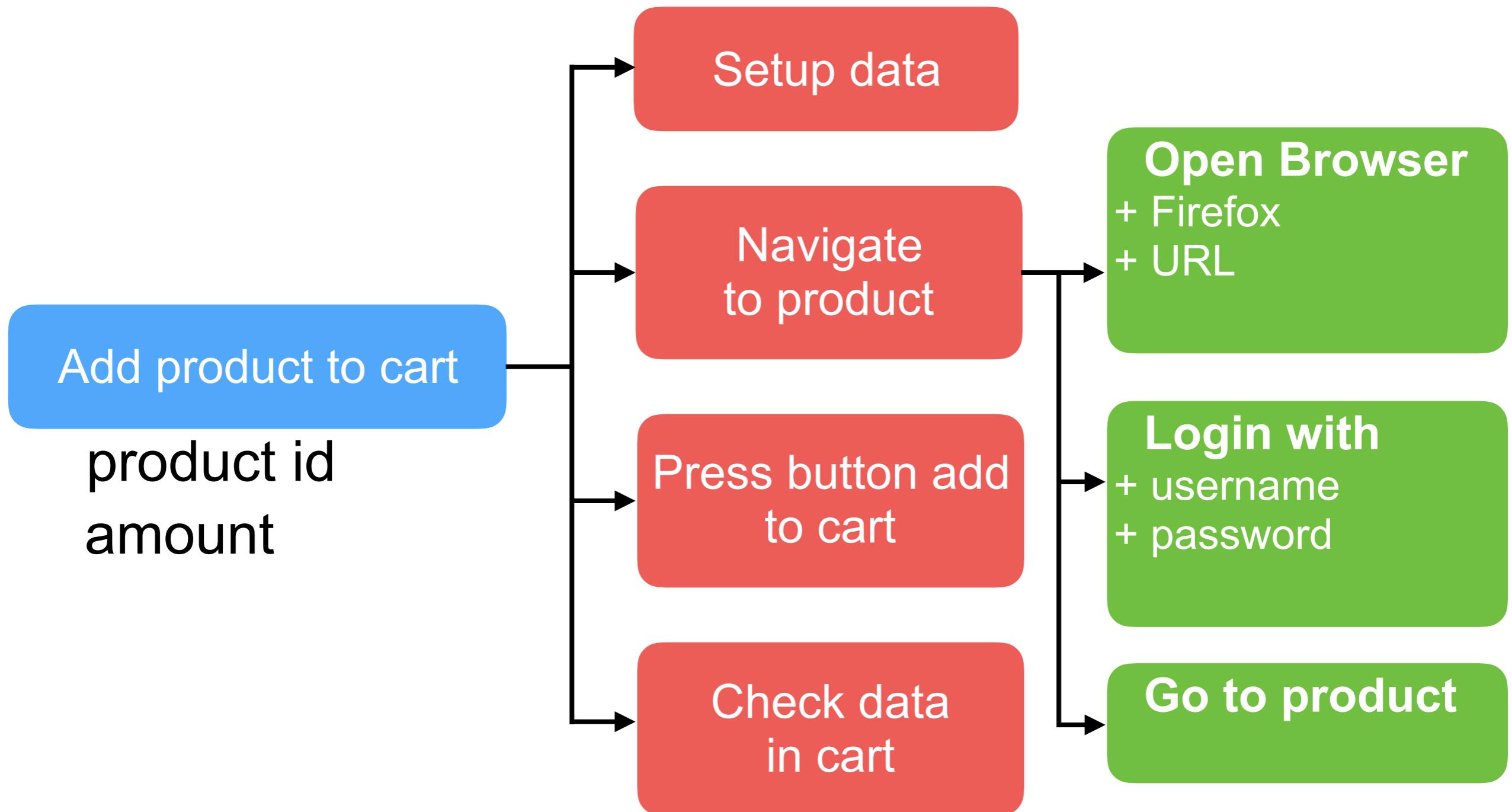
# Keyword-driven development

*“Add product to cart”*

The screenshot shows the Amazon product page for the book "Robot Framework Test Automation" by Sumit Bisht. The page includes the Amazon header with navigation links like Departments, Prime, Video, Music, and a search bar. The main content features the book's title, author, and a 4-star rating from 9 reviews. A "Look inside" button is visible. The book cover image shows a grid of mechanical components. Below the cover, the subtitle "Community Experience Distilled" and the tagline "Create test suites and automated acceptance tests from scratch" are displayed. The product details section shows Kindle (\$10.52), Paperback (\$29.99), and Other Sellers (from \$23.44). The Paperback option is highlighted. The "Buy new" section indicates the item is in stock and ships from Amazon.com with gift-wrap available. A note specifies shipping to Samsennai, Thailand, with options for 15 hrs 29 mins via AmazonGlobal Priority Shipping. The "Add to Cart" button is prominently displayed at the bottom right.



# Add product to cart



# Working with web browser



# Working with Web Browser

Robot Framework IDE/Editor



Robot Framework

Build-in  
library

Selenium  
library

Faker  
library

Python

PIP



Web Driver (ChromeDriver, GeckoDriver, IEDriver)

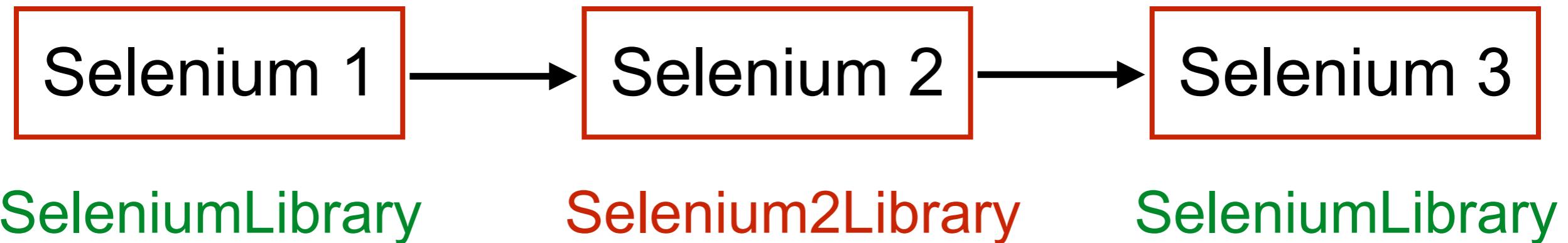


Web Browser (Google Chrome, Firefox, IE)



# Install SeleniumLibrary

\$pip install robotframework-seleniumlibrary



<https://github.com/robotframework/SeleniumLibrary/>



# SeleniumLibrary keywords

**C**heckbox Should Be Selected · **C**heckbox Should Not Be Selected · Choose File · Clear Element Text · Click Button · Click Element · Click Element At Coordinates · Click Image · Click Link · Close All Browsers · Close Browser · Close Window · Cover Element · Create Webdriver · Current Frame Should Contain · Current Frame Should Not Contain · Delete All Cookies · Delete Cookie · Double Click Element · Drag And Drop · Drag And Drop By Offset · Element Attribute Value Should Be · Element Should Be Disabled · Element Should Be Enabled · Element Should Be Focused · Element Should Be Visible · Element Should Contain · Element Should Not Be Visible · Element Should Not Contain · Element Text Should Be · Element Text Should Not Be · Execute Async Javascript · Execute Javascript · Frame Should Contain · Get All Links · Get Browser Aliases · Get Browser Ids · Get Cookie · Get Cookies · Get Element Attribute · Get Element Count · Get Element Size · Get Horizontal Position · Get List Items · Get Location · Get Locations · Get Selected List Label · Get Selected List Labels · Get Selected List Value · Get Selected List Values · Get Selenium Implicit Wait · Get Selenium Speed · Get Selenium Timeout · Get Session Id · Get Source · Get Table Cell · Get Text · Get Title · Get Value · Get Vertical Position · Get WebElement · Get WebElements · Get Window Handles · Get Window Identifiers · Get Window Names · Get Window Position · Get Window Size · Get Window Titles · Go Back · Go To · Handle Alert · Input Password · Input Text · Input Text Into Alert · List Selection Should Be · List Should Have No Selections · Location Should Be · Location Should Contain · Locator Should Match X Times · Log Location · Log Source · Log Title · Maximize Browser Window · Mouse Down · Mouse Down On Image · Mouse Down On Link · Mouse Out · Mouse Over · Mouse Up · Open Browser · Open Context Menu · Page Should Contain · Page Should Contain Button · Page Should Contain Checkbox · Page Should Contain Element · Page Should Contain Image · Page Should Contain Link · Page Should Contain List · Page Should Contain Radio Button · Page Should Contain Textfield · Page Should Not Contain · Page Should Not Contain Button · Page Should Not Contain Checkbox · Page Should Not Contain Element · Page Should Not Contain Image · Page Should Not Contain Link · Page Should Not Contain List · Page Should Not Contain Radio Button · Page Should Not Contain Textfield · Press Key · Press Keys · Radio Button Should Be Set To · Radio Button Should Not Be Selected · Register Keyword To Run On Failure · Reload Page · Remove Location Strategy · Scroll Element Into View · Select All From List · Select Checkbox · Select Frame · Select From List By Index · Select From List By Label · Select From List By Value · Select Radio Button · Select Window · Set Browser Implicit Wait · Set Focus To Element · Set Screenshot Directory · Set Selenium Implicit Wait · Set Selenium Speed · Set Selenium Timeout · Set Window Position · Set Window Size · Simulate Event · Submit Form · Switch Browser · Switch Window · Table Cell Should Contain · Table Column Should Contain · Table Footer Should Contain · Table Header Should Contain · Table Row Should Contain · Table Should Contain · Textarea Should Contain · Textarea Value Should Be · Textfield Should Contain · Textfield Value Should Be · Title Should Be · Unselect All From List · Unselect Checkbox · Unselect Frame · Unselect From List By Index · Unselect From List By Label · Unselect From List By Value · Wait For Condition · Wait Until Element Contains · Wait Until Element Does Not Contain · Wait Until Element Is Enabled · Wait Until Element Is Not Visible · Wait Until Element Is Visible · Wait Until Location Contains · Wait Until Location Is · Wait Until Page Contains · Wait Until Page Contains Element · Wait Until Page Does Not Contain · Wait Until Page Does Not Contain Element

<https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html>



# Open Browser

Browser	Name(s)
Firefox	firefox, ff
Google Chrome	googlechrome, chrome, gc
Headless Firefox	headlessfirefox
Headless Chrome	headlesschrome
Internet Explorer	internetexplorer, ie
Edge	edge
Safari	safari
Opera	opera
Android	android
Iphone	iphone
PhantomJS	phantomjs
HTMLUnit	htmlunit
HTMLUnit with Javascript	htmlunitwithjs

<https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html>



# Locator strategies

Strategy	Match based on	Example
id	Element <code>id</code> .	<code>id:example</code>
name	<code>name</code> attribute.	<code>name:example</code>
identifier	Either <code>id</code> or <code>name</code> .	<code>identifier:example</code>
class	Element <code>class</code> .	<code>class:example</code>
tag	Tag name.	<code>tag:div</code>
xpath	XPath expression.	<code>xpath://div[@id="example"]</code>
css	CSS selector.	<code>css:div#example</code>
dom	DOM expression.	<code>dom:document.images[5]</code>
link	Exact text a link has.	<code>link:The example</code>
partial link	Partial link text.	<code>partial link:he ex</code>
sizzle	Sizzle selector deprecated.	<code>sizzle:div.example</code>
jquery	jQuery expression.	<code>jquery:div.example</code>
default	Keyword specific default behavior.	<code>default:example</code>

<https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html>



# Verify elements in browser

Using keyword that have prefix “Wait Until”

<b>Wait Until Element Contains</b>	<code>locator, text, timeout=None, error=None</code>	Waits until the element <code>locator</code> contains <code>text</code> . Fails if <code>timeout</code> expires before the text appears. See the <i>Timeouts</i> section for details about the locator syntax. <code>error</code> can be used to override the default error message.
<b>Wait Until Element Does Not Contain</b>	<code>locator, text, timeout=None, error=None</code>	Waits until the element <code>locator</code> does not contain <code>text</code> . Fails if <code>timeout</code> expires before the text disappears. See the <i>Timeouts</i> section for details about the locator syntax. <code>error</code> can be used to override the default error message.
<b>Wait Until Element Is Enabled</b>	<code>locator, timeout=None, error=None</code>	Waits until the element <code>locator</code> is enabled. Element is considered enabled if it is not disabled nor read-only. Fails if <code>timeout</code> expires before the element is enabled. See the <i>Timeouts</i> section for details about the locator syntax. <code>error</code> can be used to override the default error message. Considering read-only elements to be disabled is a new feature in Selenium 4.
<b>Wait Until Element Is Not Visible</b>	<code>locator, timeout=None, error=None</code>	Waits until the element <code>locator</code> is not visible. Fails if <code>timeout</code> expires before the element is not visible. See the <i>Timeouts</i> section for details about the locator syntax.

<https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html>



# First test case

Create file => **google\_success.robot**

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

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

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

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



# Run test and see result

```
$robot google_success.robot
```



# Working with Selenium

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

Library      SeleniumLibrary

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

\${URL}    http://www.google.com

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

ค้นหาคำว่า เหล็กไฟล

[Tags] done

เข้าไปยังหน้าค้นหาของ google

ค้นหาคำว่า "เหล็กไฟล"

จะต้องเจอ "เหล็กไฟล" นะ

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

ค้นหาคำว่า "\${keyword}"

Input Text               name:q      \${keyword}

Press Keys               name:q      RETURN



# Setup web browser driver

Web Browser	Driver
Google Chrome	ChromeDriver
FireFox	GeckoDriver
Internet Explorer	InternetExplorerDriver
Microsoft Edge	EdgeDriver



# eg. ChromeDriver

```
$set WEB_DRIVER=<path of driver>
$set PATH=.;%WEB_DRIVER%;%PATH%
```

<https://chromedriver.chromium.org/downloads>



# Reporting

## Open file report.html

### Comments Api Report

Generated  
20190925 16:13:40 UTC+07:00  
13 days 20 hours ago

#### Summary Information

Status: All tests passed  
Start Time: 20190925 16:13:40.553  
End Time: 20190925 16:13:40.961  
Elapsed Time: 00:00:00.408  
Log File: log.html

#### Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:00	<div style="width: 100%; background-color: green;"></div>
All Tests	1	1	0	00:00:00	<div style="width: 100%; background-color: green;"></div>

Statistics by Tag

Statistics by Tag	Total
No Tags	

Statistics by Suite

Statistics by Suite	Total
Comments Api	1

#### Test Details

Totals Tags Suites Search

Type:  Critical Tests  All Tests

### Comments Api Log

Generated  
20190925 16:13:40 UTC+07:00  
13 days 20 hours ago

#### Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	1	1	0	00:00:00	<div style="width: 100%; background-color: green;"></div>
All Tests	1	1	0	00:00:00	<div style="width: 100%; background-color: green;"></div>

Statistics by Tag

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Comments Api	1	1	0	00:00:00	<div style="width: 100%; background-color: green;"></div>

#### Test Execution Log

- **SUITE** Comments Api

Full Name: Comments Api  
Source: /Users/somkiat/data/slide/robotframework/aycap-20190924/comments\_api.robot  
Start / End / Elapsed: 20190925 16:13:40.553 / 20190925 16:13:40.961 / 00:00:00.408  
Status: 1 critical test, 1 passed, 0 failed  
1 test total, 1 passed, 0 failed

- **TEST** Flow 1

Full Name: Comments Api.Flow 1  
Start / End / Elapsed: 20190925 16:13:40.750 / 20190925 16:13:40.960 / 00:00:00.210  
Status: PASS (critical)

+ KEYWORD \${email} = Get Comments  
+ KEYWORD BuiltIn.Log To Console \${email}

Automated testing  
© 2017 - 2018 Siam Chamnkit Company Limited. All rights reserved.

159

# Variables

Define global variables in test suites

```
*** Variables ***
${URL}  http://www.google.com
```



# Variables

Set value of variables from command line

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

**URL**

http://www.google.com



\$robot -v URL:<value> google\_success.robot



# Build-ins variables

Usage	Description
<code> \${SPACE}</code>	\ \
<code>"\${SPACE}"</code>	" "
<code> \${SPACE * 5}</code>	\ \ \ \ \
<code>"\${SPACE * 5}"</code>	" \ \ \ \ "
<code> \${EMPTY}</code>	\



# Tagging test cases

## Grouping your test cases

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

ค้นหาคำว่า robot

[Tags] done

ค้นหาคำว่า robot

ต้องเจอคำว่า robot

ค้นหาคำว่า thai

[Tags] testing

ค้นหาคำว่า thai

ต้องเจอคำว่า robot



# Run tests with specified tags

\$robot -i testing google\_success.robot

\$robot -e done google\_success.robot

*-i = Include, -e = Exclude*



# User keyword arguments

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

ค้นหาคำว่า robot

ค้นหาคำว่า

ต้องเจอกำคำว่า robot

robot

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

ค้นหาคำว่า

[Arguments]

name:q \${keyword}

Input Text

name:q \${keyword}

Press Keys

name:q RETURN



# Embedding arguments

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

ค้นหาคำว่า robot

ค้นหาคำว่า “robot”

ต้องเจอกำกว่า r~~obot~~bot

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

ค้นหาคำว่า “\${keyword}”

Input Text name:q \${keyword}

Press Keys name:q RETURN



# Improve your test with Test Life Cycle



# Test Life Cycle

Suite

Suite Setup

Test Setup  
Test case 1  
Test Teardown

Test Setup  
Test case 2  
Test Teardown

Test Setup  
Test case 3  
Test Teardown

Suite Teardown



# Test Life Cycle

## \*\*\* Settings \*\*\*

Suite Setup

เข้า web google

Test Setup

เพิ่มข้อมูลการทดสอบ

Test Teardown

กลับไปยังหน้าแรก

Suite Teardown

Close Browser



# Improve your test with Test Template



# Test template

One template for all test cases

```
*** Settings ***
Test Template    Search ${keyword} should be found ${expected result}
```

```
*** Test Cases ***
```

Case 01	robot	robot
Case 02	thai	thai

```
*** Keywords ***
```

```
Search ${keyword} should be found ${expected result}
```

ค้นหาคำว่า \${keyword}

ต้องเจอคำว่า robot



# Test template

## Template per test case

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

Search from google

[Template] Flow success to search from google

robot                robot

thai                thai

### \*\*\* Keywords \*\*\*

Flow success to search from google

[Arguments] \${keyword}    \${expected result}

Search \${keyword} shoud be found \${expected result}



# Reuse keywords/variables with Resource



# Using resource

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

Resource resources/sample.robot

Suite Setup sample.เข้า web google

Suite Teardown Close Browser

1. Create folder = resources
2. Create file = resources/sample.robot

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

Library SeleniumLibrary

Library String

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

\\${URL} http://www.google.com

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

กลับไปยังหน้าแรก

Go To \\${URL}



# Page Object Pattern



# Problems

Fragile tests

Bad maintainability

Hard to develop new tests

Hard to understand

Test code duplication

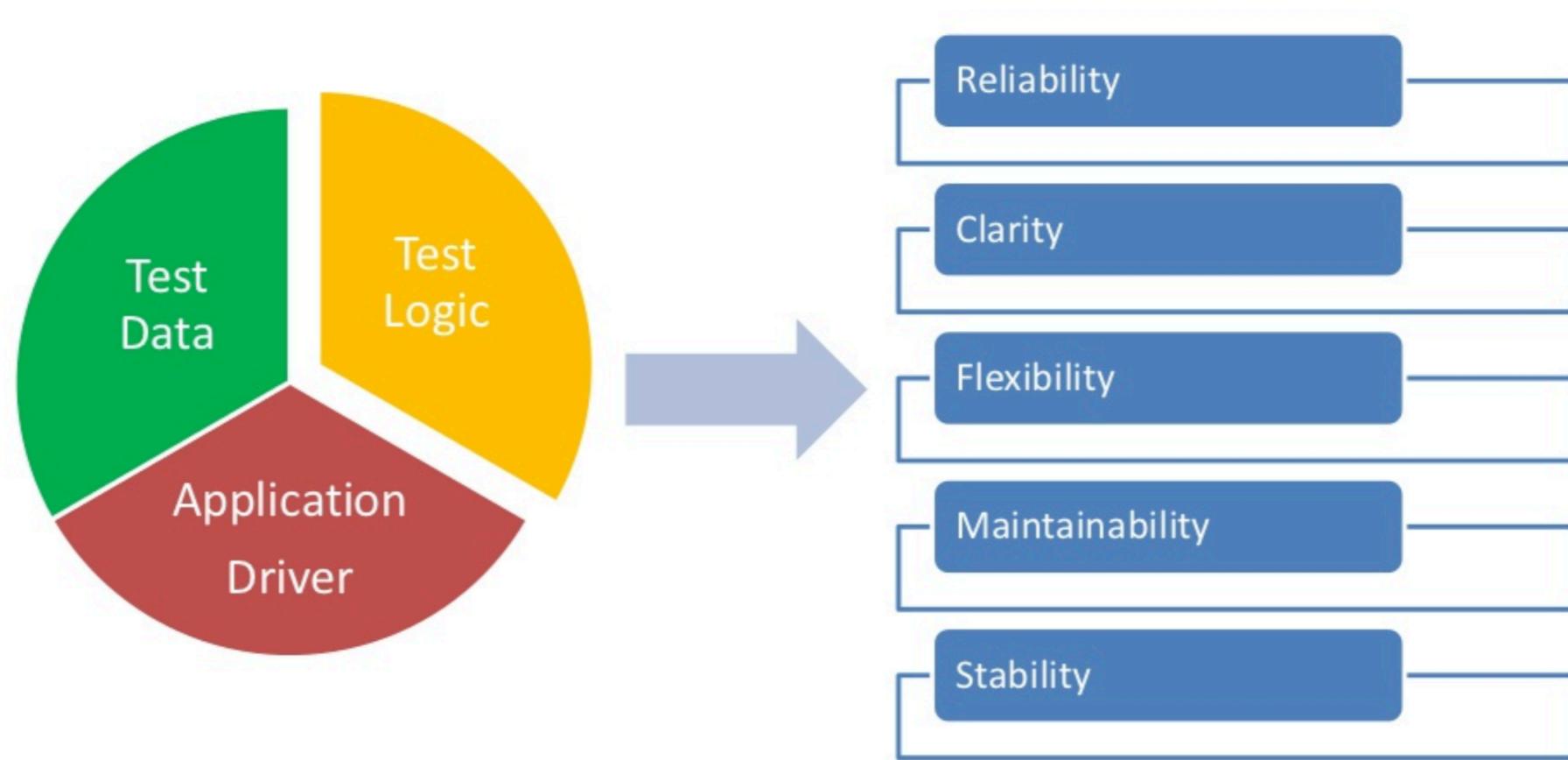
High cost to maintenance tests



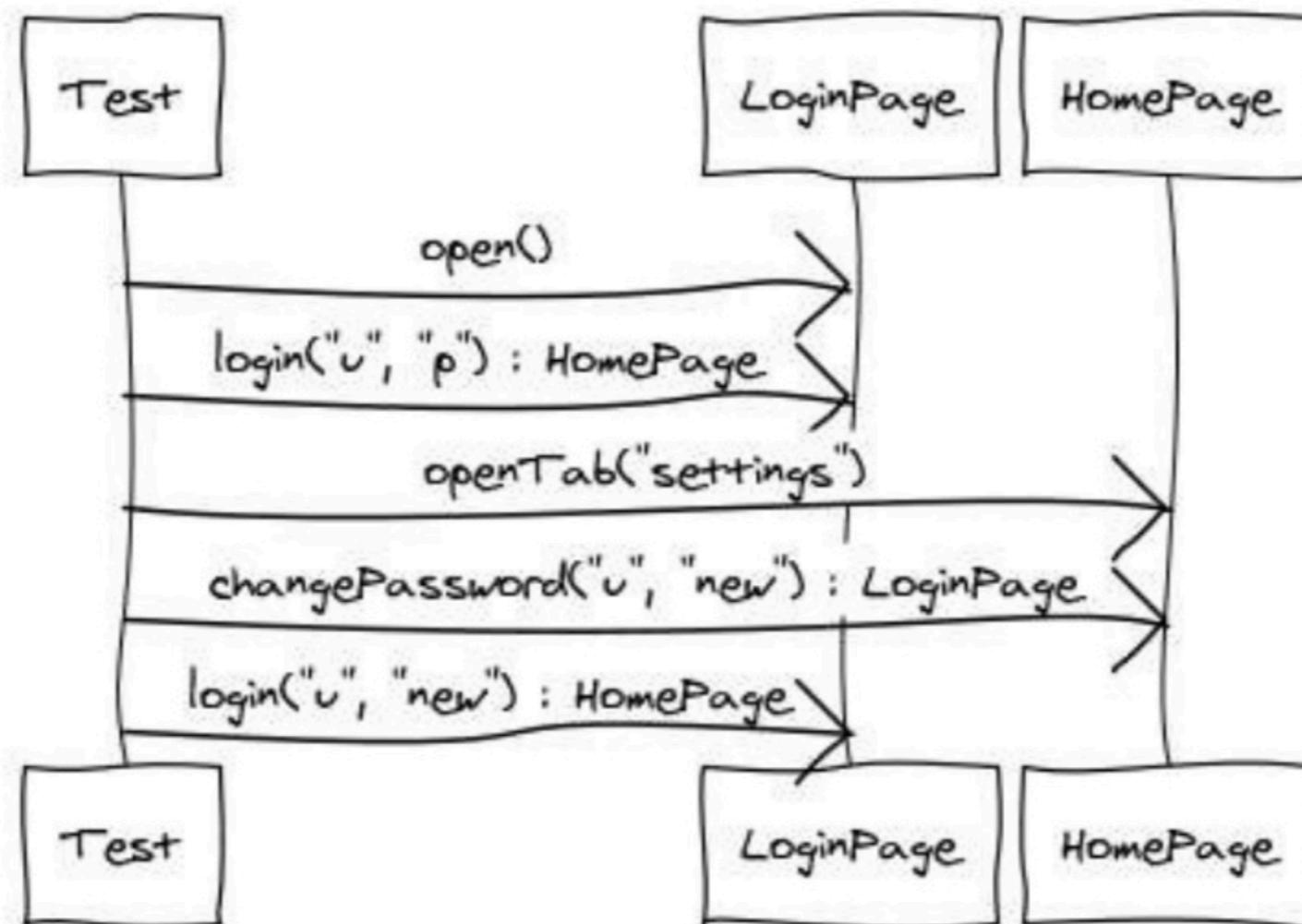
# Primary goal ?

To enable reliable stable tests

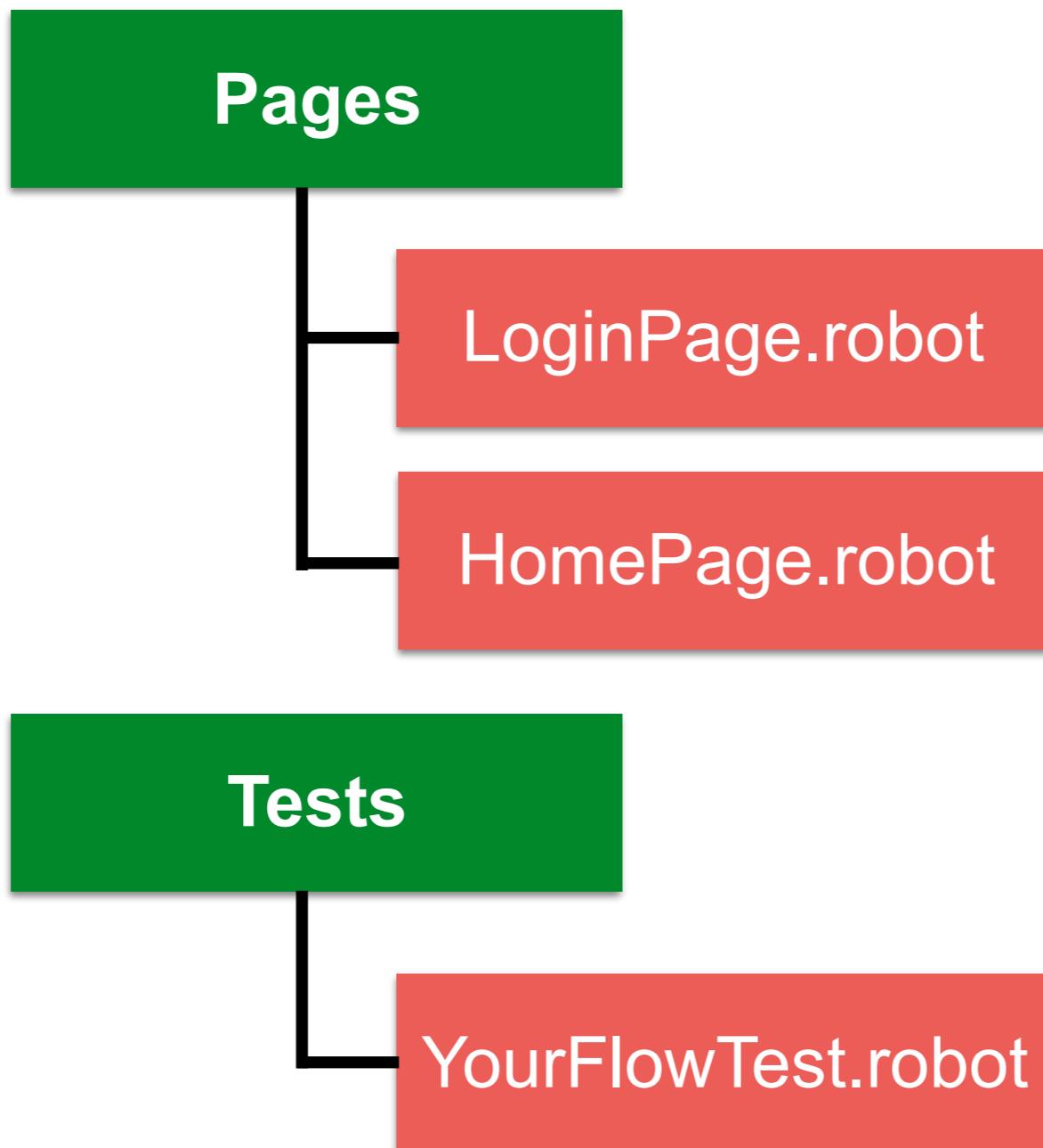
We need to separate all parts as much as possible



# Test architecture with Page object



# Test architecture with Page object



# Tips and Tricks

Basic page with browser details

Expose only what is allowed to do on page

Don't use browser details in tests

Use site map to prepare Page object



# Try to improve ...

Reliable tests

Reusable test code

Separation of concepts

Expressive UI structure

Short and clear tests

Tests look more like acceptance tests

Tests are understand by non-technical people

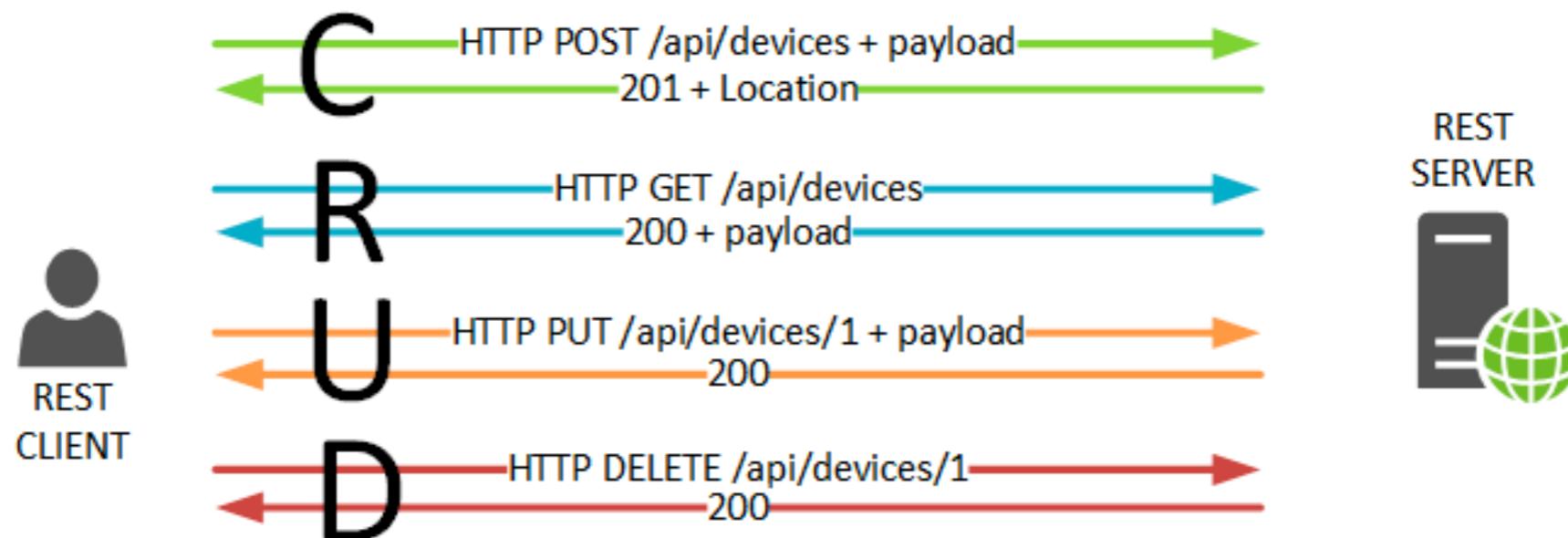


# API Testing



# Install RequestsLibrary

\$pip install requests  
\$pip install robotframework-requests



<https://github.com/bulkan/robotframework-requests/>



# RequestsLibrary

## Introduction

RequestsLibrary is a [Robot Framework](#) test library that uses the [Requests](#) HTTP client.

Here is an example testcase

*** Settings ***				
Library	Collections			
Library	RequestsLibrary			
*** Test Cases ***				
Get Requests				
	Create Session	github	<a href="http://api.github.com">http://api.github.com</a>	
	Create Session	google	<a href="http://www.google.com">http://www.google.com</a>	
	\${resp}=	Get Request	google	/
	Should Be Equal As Strings	\${resp.status_code}	200	
	\${resp}=	Get Request	github	/users/bulkan
	Should Be Equal As Strings	\${resp.status_code}	200	
	Dictionary Should Contain Value	\${resp.json()}	Bulkan Savun Evcimen	

## Shortcuts

[Create Digest Session](#) · [Create Ntlm Session](#) · [Create Session](#) · [Delete](#) · [Delete All Sessions](#) · [Delete Request](#) · [Get](#) · [Get Request](#) · [Head](#) · [Head Request](#) · [Options](#) · [Options Request](#) · [Patch](#) · [Patch Request](#) · [Post](#) · [Post Request](#) · [Put](#) · [Put Request](#) · [To Json](#)

<https://bulkan.github.io/robotframework-requests/doc/RequestsLibrary.html>



# RequestsLibrary keywords

## Shortcuts

[Create Client Cert Session](#) · [Create Custom Session](#) · [Create Digest Session](#) · [Create Ntlm Session](#) ·  
[Create Session](#) · [Delete All Sessions](#) · [Delete Request](#) · [Get Request](#) · [Head Request](#) · [Options Request](#) ·  
[Patch Request](#) · [Post Request](#) · [Put Request](#) · [Session Exists](#) · [To Json](#) · [Update Session](#)

<https://bulkan.github.io/robotframework-requests/doc/RequestsLibrary.html>



# Create tests

Connect to service and check status code

```
*** Settings ***
```

```
Library RequestsLibrary
```

```
*** Test Case ***
```

```
Success with /users
```

```
Create Session jph https://jsonplaceholder.typicode.com
```

```
    ${response}= Get Request jph /users
```

```
Should Be Equal ${response.status_code} ${200}
```



# Create tests

## Check size of result from service

\*\*\* Test Case \*\*\*

Success with /users

```
Create Session    jph  https://jsonplaceholder.typicode.com
${response}=  Get Request  jph  /users
Should Be Equal  ${response.status_code}  ${200}
```

```
 ${json}=  Set Variable  ${response.json()}
Length Should Be  ${json}  10
```



# Create tests

## Check first user data from result

\*\*\* Test Case \*\*\*

Success with /users

```
Create Session    jph  https://jsonplaceholder.typicode.com
${response}=  Get Request  jph  /users
Should Be Equal  ${response.status_code}  ${200}
```

```
${json}=  Set Variable  ${response.json()}
Length Should Be  ${json}  10
```

```
Should Be Equal  ${json[0]["name"]}  Leanne Graham
Should Be Equal  ${json[0]["username"]}  Bret
```



# Working with HTTP POST

Add product id=1000 to empty basket

Create Session baskets <http://localhost:8882>

&{headers}= Create Dictionary Content-Type=application/json

&{data}= Create Dictionary

... product\_id=\${1000}

... product\_name=Adidas

... product\_price=\${1500}

... product\_image=<http://xxx.jpg>

... quantity=\${1}

\${response}= Post Request baskets /baskets

... data=\${data} headers=\${headers}

Should Be Equal As Strings \${response.status\_code} 200



# Parallel testing with pabot



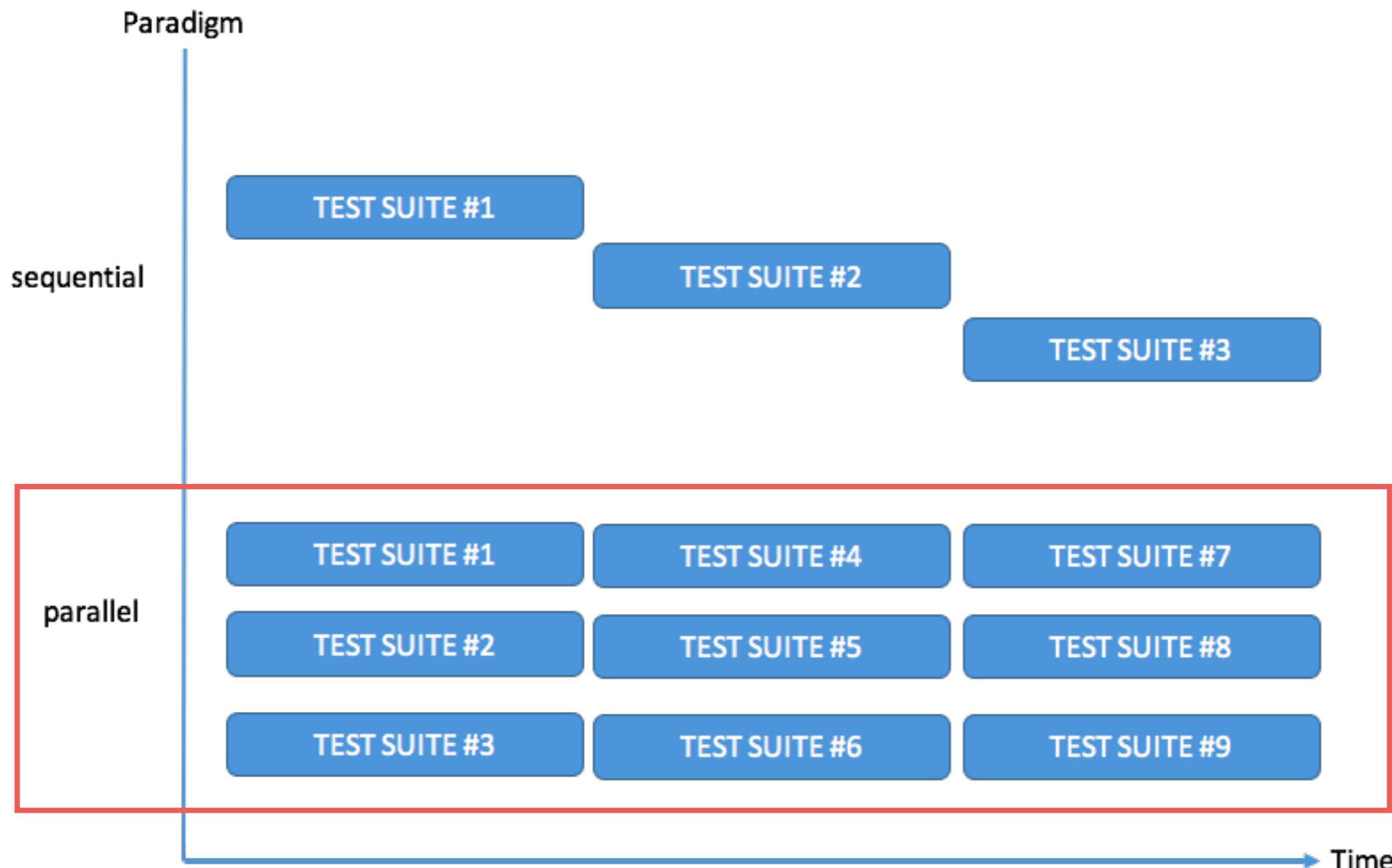
# Problems

More test cases more problems

More test cases more testing time

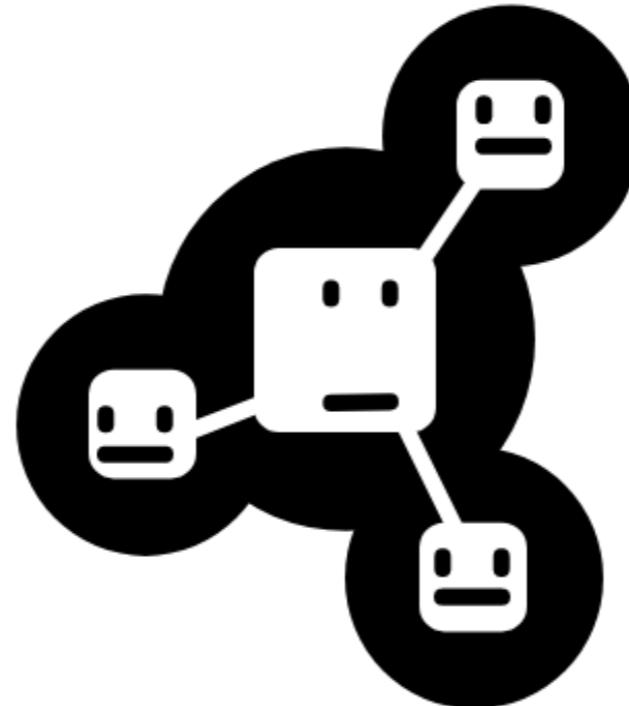


# Parallel testing



# Install Pabot

```
$pip install robotframework-pabot  
$pabot
```



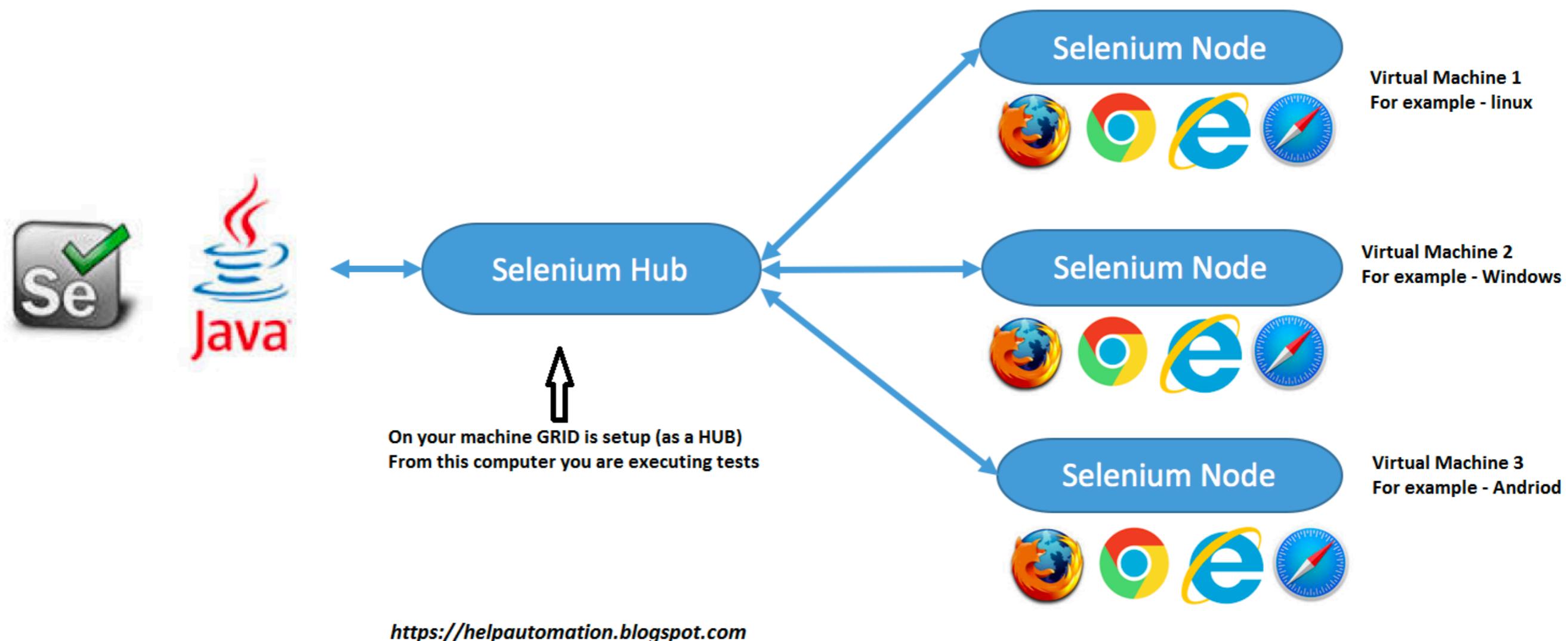
<https://pabot.org/>



# Scaling UI testing with Selenium Grid



# Selenium Grid



<https://github.com/SeleniumHQ/selenium/wiki/Grid2>



# Using from Robot Framework

```
Open Browser    http://localhost:8080/demo/
...  browser=chrome
...  remote_url=http://localhost:4444/wd/hub
...  desired_capabilities=browserName:chrome
```



# Continuous Testing



<https://jenkins.io/>



# 1. Create new job

Enter an item name  
**Fill in job name**  
» Required field

 **Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

 **Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

 **Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

 **Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

 **GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

 **Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

**OK**



# 2. Source code management

General    **Source Code Management**    Build Triggers    Build Environment    Build    Post-build Actions

## Source Code Management

None  
 Git  
 Subversion

(?)

## Build Triggers

Trigger builds remotely (e.g., from scripts) (?)  
 Build after other projects are built (?)  
 Build periodically (?)  
 GitHub hook trigger for GITScm polling (?)  
 Poll SCM (?)



# 3. Build trigger

General    **Source Code Management**    Build Triggers    Build Environment    Build    Post-build Actions

---

## Source Code Management

None  
 Git  
 Subversion ?

---

## Build Triggers

Trigger builds remotely (e.g., from scripts) ?  
 Build after other projects are built ?  
 Build periodically ?  
 GitHub hook trigger for GITScm polling ?  
 Poll SCM ?



# 4. Build trigger with poll SCM

Poll SCM ?

Schedule

\* \* \* \* \*

No schedules so will only run due to SCM changes if triggered by a post-commit hook

This field follows the syntax of cron (with minor differences). Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DOM MONTH DOW

MINUTE Minutes within the hour (0–59)

HOUR The hour of the day (0–23)

DOM The day of the month (1–31)

MONTH The month (1–12)

DOW The day of the week (0–7) where 0 and 7 are Sunday.

To specify multiple values for one field, the following operators are available. In the order of precedence,

- \* specifies all valid values
- M–N specifies a range of values
- M–N/X or \*/X steps by intervals of X through the specified range or whole valid range
- A, B, . . . , Z enumerates multiple values



# 5. Build your job

Build

Add build step ▾

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit



# 6. Post build actions

**Post-build Actions**

Add post-build action ▾

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Publish Robot Framework test results**
- Record fingerprints of files to track usage
- Git Publisher
- E-mail Notification
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Delete workspace when build is done



# 7. Add post build with Robot framework

**Post-build Actions**

**Publish Robot Framework test results**

Directory of Robot output

Path to directory containing robot xml and html files (relative to build workspace)

Advanced...

Thresholds for build result

Yellow %  **Entry must be percentage value between 0-100**

Blue %  **Entry must be percentage value between 0-100**

Use thresholds for critical tests only

Add post-build action ▾



# 8. Install Robot framework plugin

The screenshot shows the Jenkins 'Manage Jenkins' interface. On the left, there's a sidebar with links: People, Build History, Manage Jenkins (which is highlighted with a red box and has a red circle with '1' above it), My Views, Credentials, and New View. Below the sidebar are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area is titled 'Manage Jenkins' and contains several configuration options, each with an icon and a brief description. A red box highlights the 'Manage Plugins' option, which has a red circle with '2' above it and is connected by a red line to the 'Manage Jenkins' link in the sidebar. Other options include: 'Configure System' (Configure global settings and paths), 'Configure Global Security' (Secure Jenkins; define who is allowed to access/use the system), 'Configure Credentials' (Configure the credential providers and types), 'Global Tool Configuration' (Configure tools, their locations and automatic installers), 'Reload Configuration from Disk' (Discard all the loaded data in memory and reload everything from file system. Useful when you modify disk), 'Manage Plugins' (Add, remove, disable or enable plugins that can extend the functionality of Jenkins), 'System Information' (Displays various environmental information to assist trouble-shooting), and 'System Log' (System log captures output from `java.util.logging` output related to Jenkins).

1

2

## Manage Jenkins

- Configure System**  
Configure global settings and paths.
- Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.
- Configure Credentials**  
Configure the credential providers and types
- Global Tool Configuration**  
Configure tools, their locations and automatic installers.
- Reload Configuration from Disk**  
Discard all the loaded data in memory and reload everything from file system. Useful when you modify disk.
- Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- System Information**  
Displays various environmental information to assist trouble-shooting.
- System Log**  
System log captures output from `java.util.logging` output related to Jenkins.



# 8. Install Robot framework plugin

Filter:

Updates Available Installed Advanced

Install ↓	Name	Version
<a href="#">Robot Framework</a> <input type="checkbox"/> This publisher stores <a href="#">Robot Framework</a> test reports for builds and shows summaries of them in project and build views along with trend graph.		1.6.5

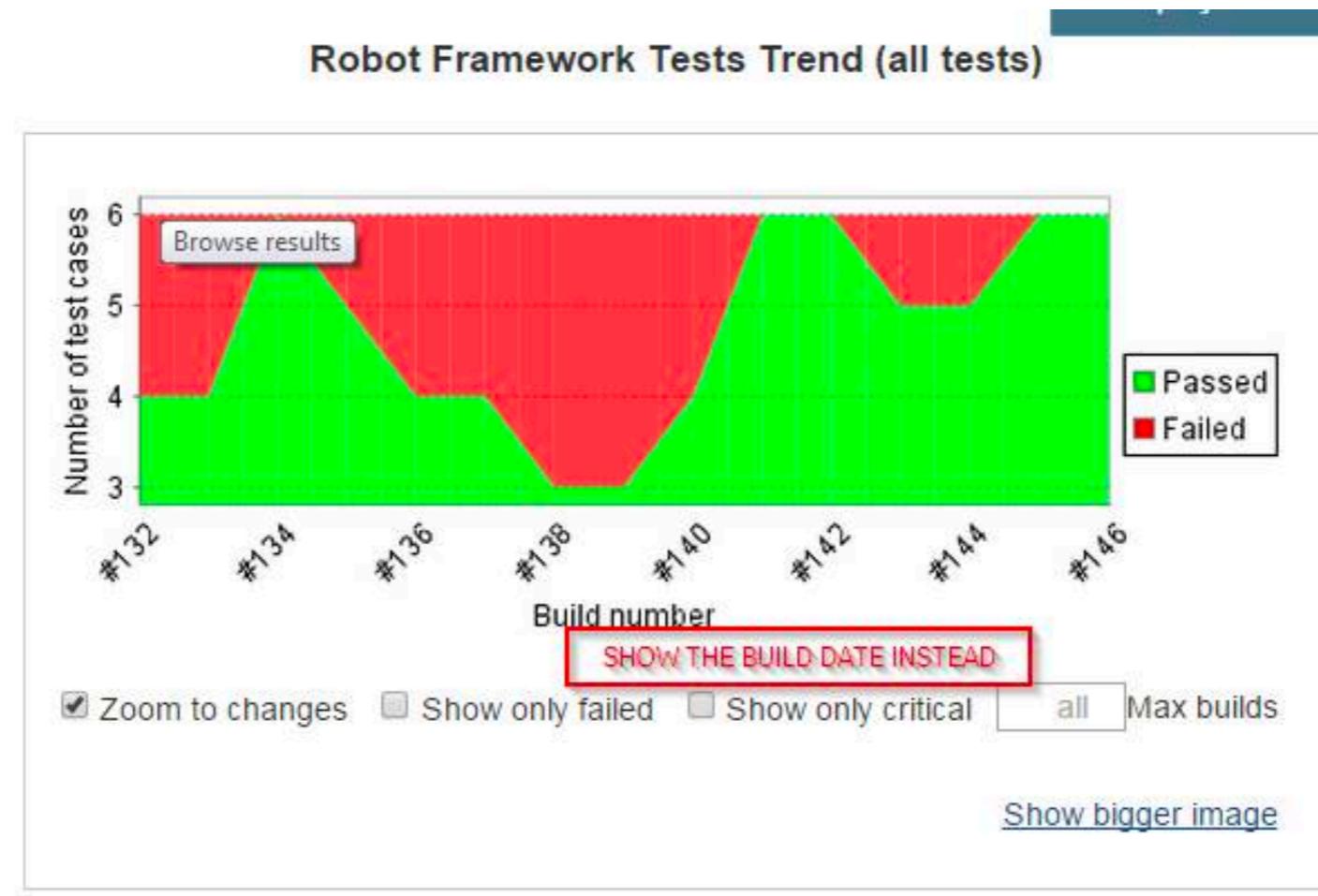
Install without restart   Download now and install after restart   Check now

Update information obtained: 3 hr 51 min ago

<https://plugins.jenkins.io/robot>



# Robot framework report



<https://plugins.jenkins.io/robot>

