

# Automation for Android

*Calaba.sh*

# Target !!

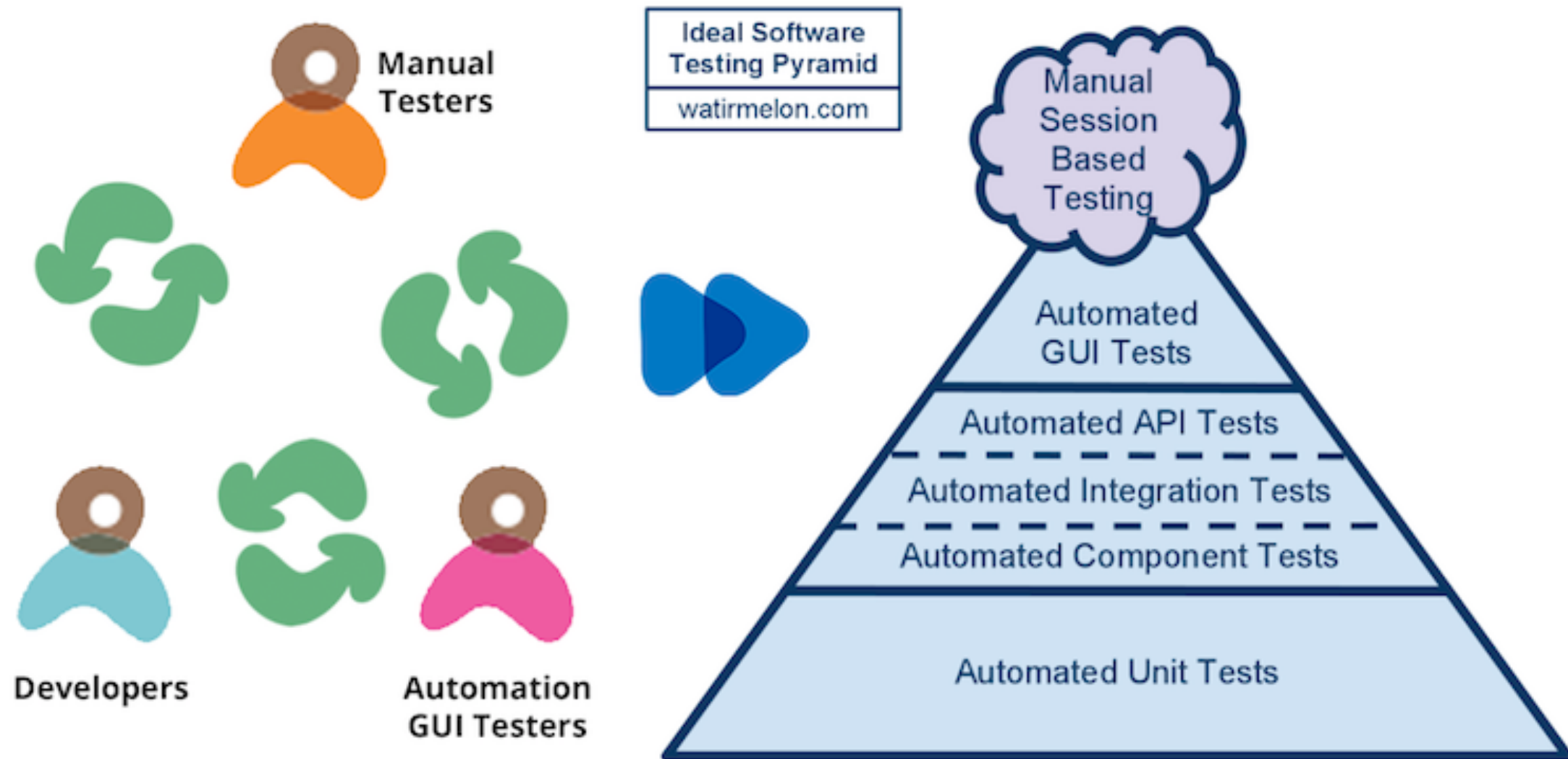
*Calaba.sh*



**Jenkins**

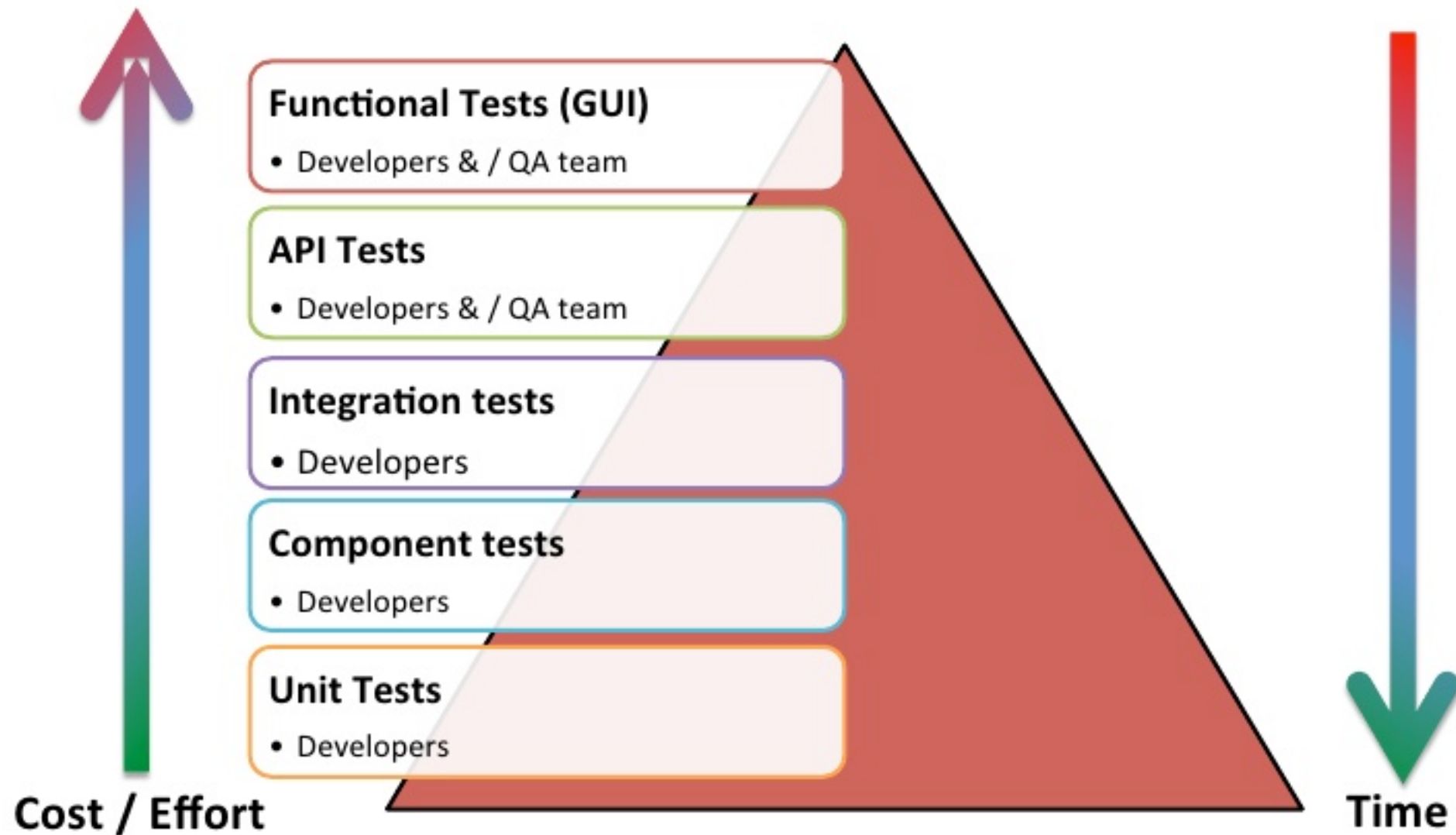


# Pyramid testing



# Pyramid testing

## Ideal Test Pyramid



# Cycle of Automation test

1. Setup data
2. Run tests
3. Validation

# Software requirement

1. Ruby 2.0 + Gem
2. Android SDK

# Installation

```
$gem install calabash-android
```

# Hello calabash





# Calabash console

\$calabash-android **console** <apk>

**Let's try ...**

# Where is APK ?



# Step 1 :: List all package

```
$adb shell pm list packages
```

# Step 2 :: Get real path

```
$adb shell pm path <com.example.someapp>
```

# Step 3:: Download apk from device

```
$adb pull </data/app/com.example.someapp.apk>
```

# Let's start



# Calabash console

\$calabash-android **console** <apk>

# Calabash console

>reinstall\_app

>start\_test\_server\_in\_background

**Let's try ...**



# Calabash console

```
>query("*")  
>query("* id:'xxxxx'")
```

**Let's try ...**

<https://github.com/calabash/calabash-android/wiki/05-Query-Syntax>

# UIAutomator Viewer

```
$cd ANDROID_HOME/tools
```

```
$/uiautomatorviewer
```

# What is Calabash ?



# What is Calabash ?

Automated acceptance test of mobile app

Cross platform (Android, iOS)

Open source

<http://calaba.sh/>

# Features

Native and hybrid app

Cross platform (Android, iOS)

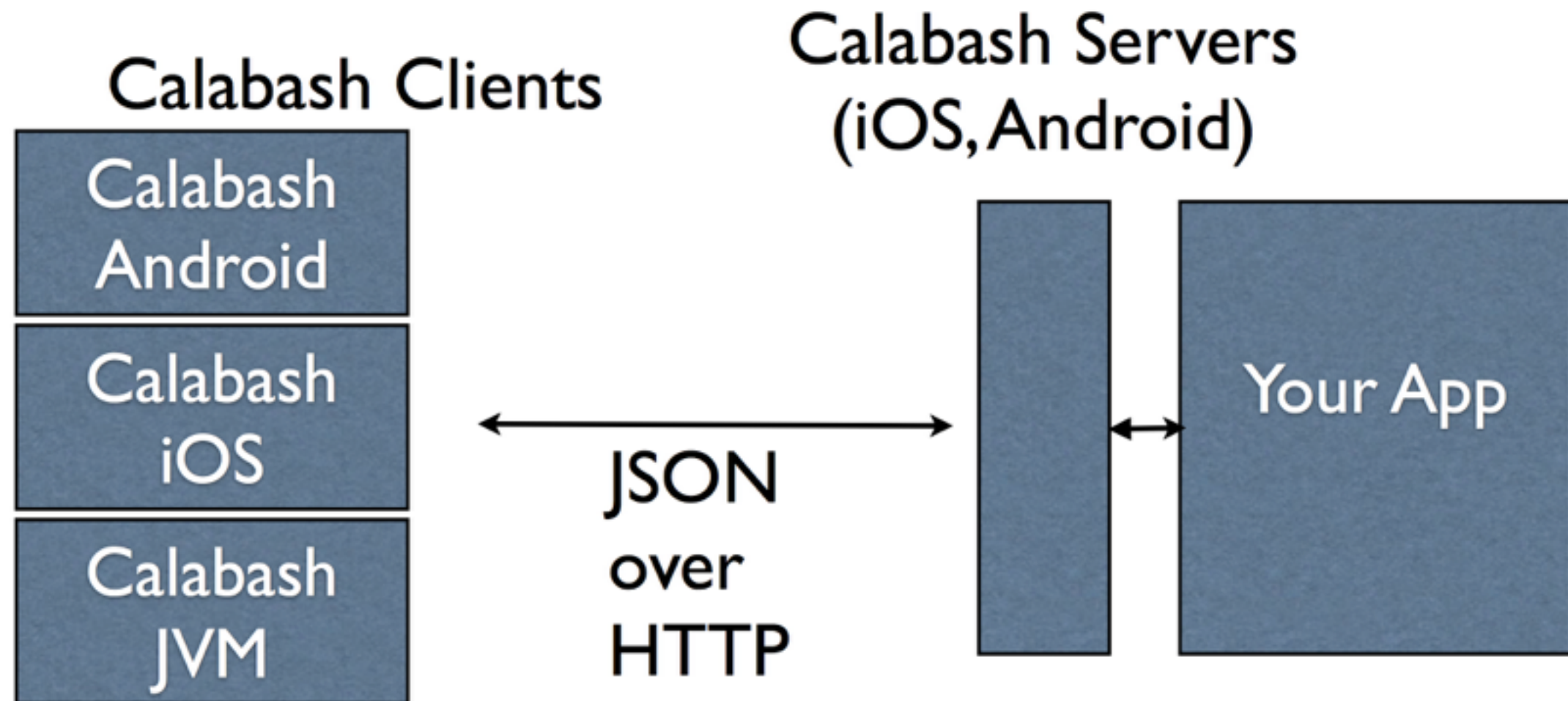
Run on physical device and emulator

<http://calaba.sh/>

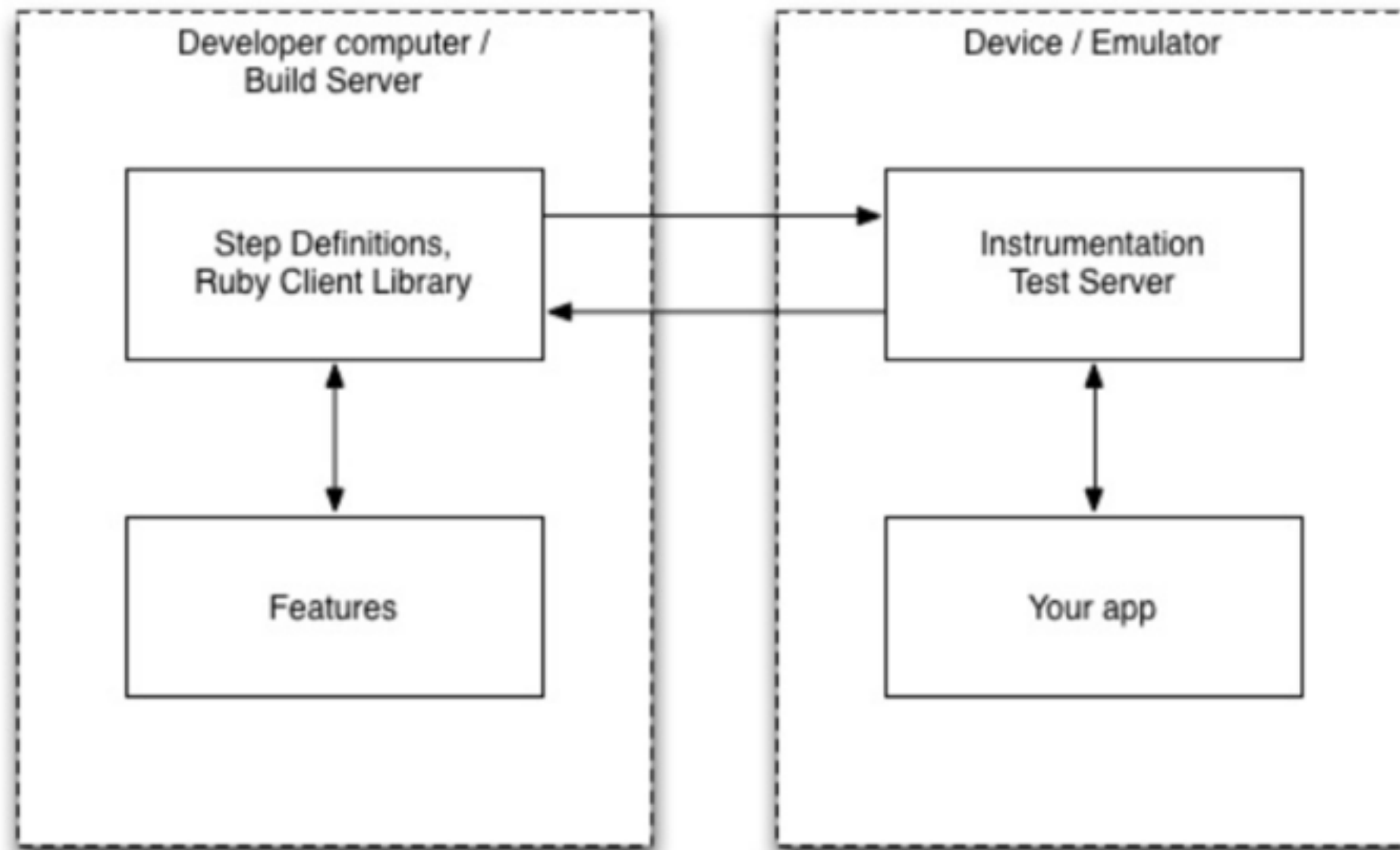
# Others ?

Flank, KIF, UIAutomation ...

# How does it work ?



# How does it work ?





# Generate project

\$ calabash-android gen

```
├── features
│   ├── my_first.feature
│   ├── step_definitions
│   │   └── calabash_steps.rb
│   └── support
│       ├── app_installation_hooks.rb
│       ├── app_life_cycle_hooks.rb
│       ├── env.rb
│       └── hooks.rb
```

# Feature

```
my_first.feature
1 Feature: Login feature
2
3 Scenario: As a valid user I can log into my app
4   When I press "Login"
5   Then I see "Welcome to coolest app ever"
```

# Given When Then

**Given** -> to put system in known state

**When** -> to describe the action

**Then** -> to observe outcomes

**And, But**

<https://github.com/cucumber/cucumber/wiki/Given-When-Then>

# Cucumber

Write in **Gherkin**

Domain Specific Language (DSL)

Business readable

Support 40 languages

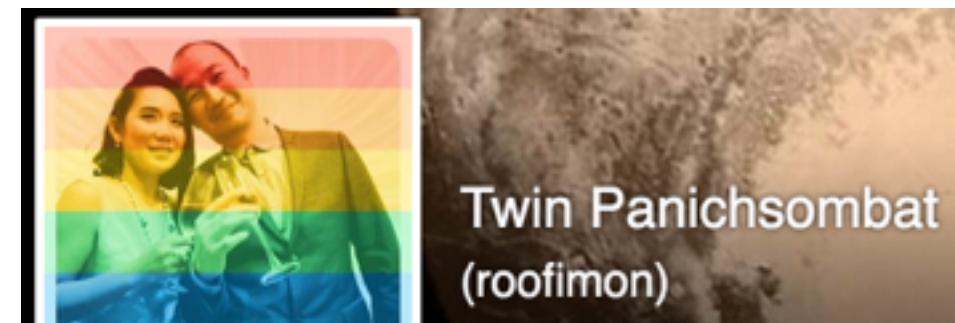
<https://github.com/cucumber/cucumber/wiki/Gherkin>

# English

```
"en": {  
  "but": "*|But",  
  "and": "*|And",  
  "then": "*|Then",  
  "when": "*|When",  
  "name": "English",  
  "native": "English",  
  "feature": "Feature|Business Need|Ability",  
  "background": "Background",  
  "scenario": "Scenario",  
  "scenario_outline": "Scenario Outline|Scenario Template",  
  "examples": "Examples|Scenarios",  
  "given": "*|Given"  
},
```

# Support Thai

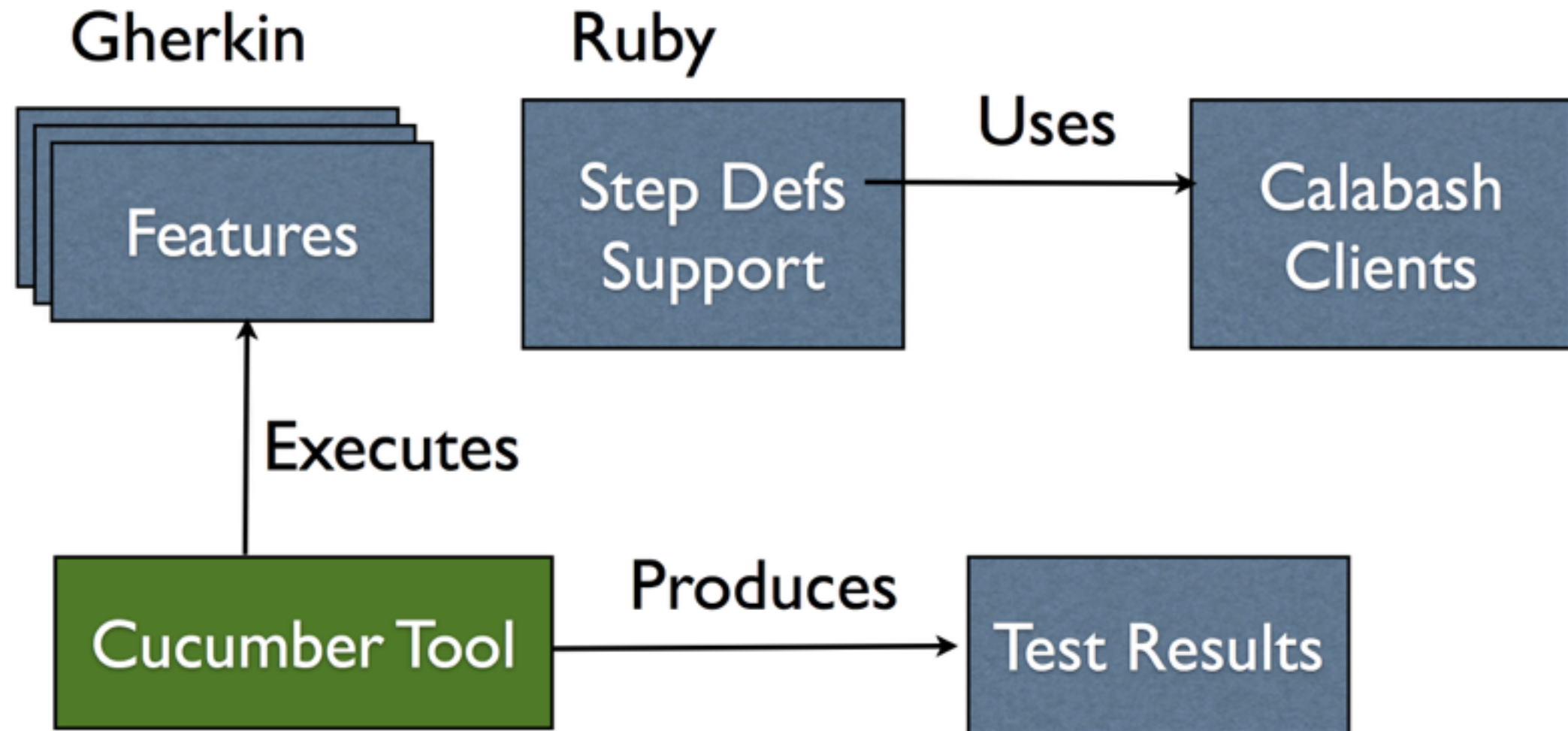
```
"th": {  
  "but": "*|แต่",  
  "and": "*|และ",  
  "then": "*|ดังนั้น",  
  "when": "*|เมื่อ",  
  "name": "Thai",  
  "native": "ไทย",  
  "feature": "โครงหลัก|ความต้องการทางธุรกิจ|ความสามารถ",  
  "background": "แนวคิด",  
  "scenario": "เหตุการณ์",  
  "scenario_outline": "สรุปเหตุการณ์|โครงสร้างของเหตุการณ์",  
  "examples": "ชุดของตัวอย่าง|ชุดของเหตุการณ์",  
  "given": "*|กำหนดให้",  
},
```



# Step Definitions

Define in **features/step\_definitions/\*\_steps.rb**  
Write in Ruby language

# Cucumber + Calabash





# Layer of cucumber

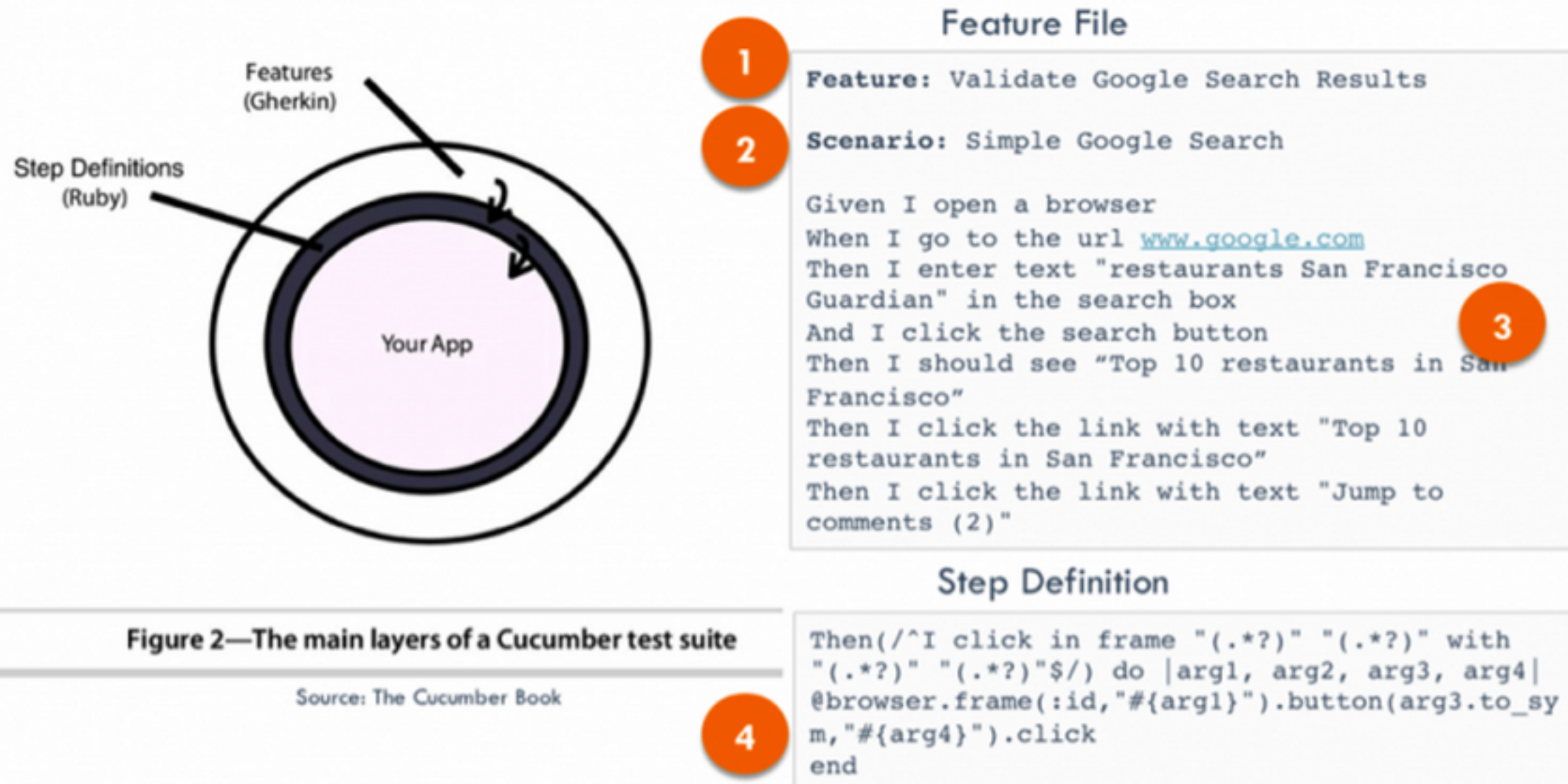
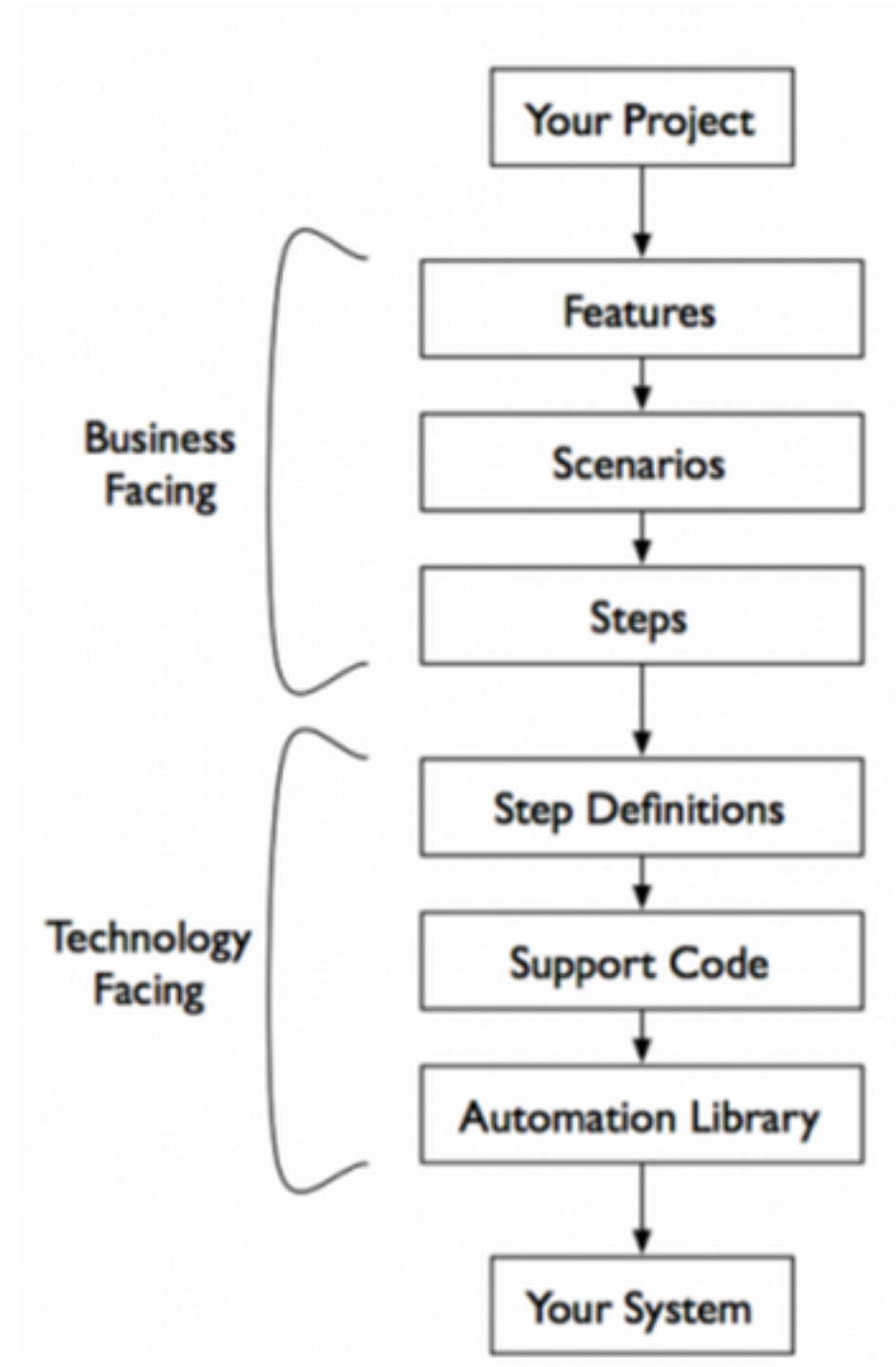


Figure 2—The main layers of a Cucumber test suite

Source: The Cucumber Book

# Cucumber stack



# Write first feature



# Run calabash

```
$ calabash-android run <APK file>
```

# Generate HTML report

```
$ calabash-android run <APK file>  
--format html --out report.html --format pretty
```

## Cucumber Features

### Feature: Login feature

Scenario: As a valid user I can log into my app

Timeout waiting for elements: \* marked:'Login'

When I press "Login"

Timeout waiting for elements: \* marked:'Login' (Calabash::Android::WaitHelpers::WaitError)

features/my\_first.feature:4:in `When I press "Login"'

```
2  
3   Scenario: As a valid user I can log into my app  
4     When I press "Login"  
5     Then I see "Welcome to coolest app ever"  
6     # gem install syntax to get syntax highlighting
```

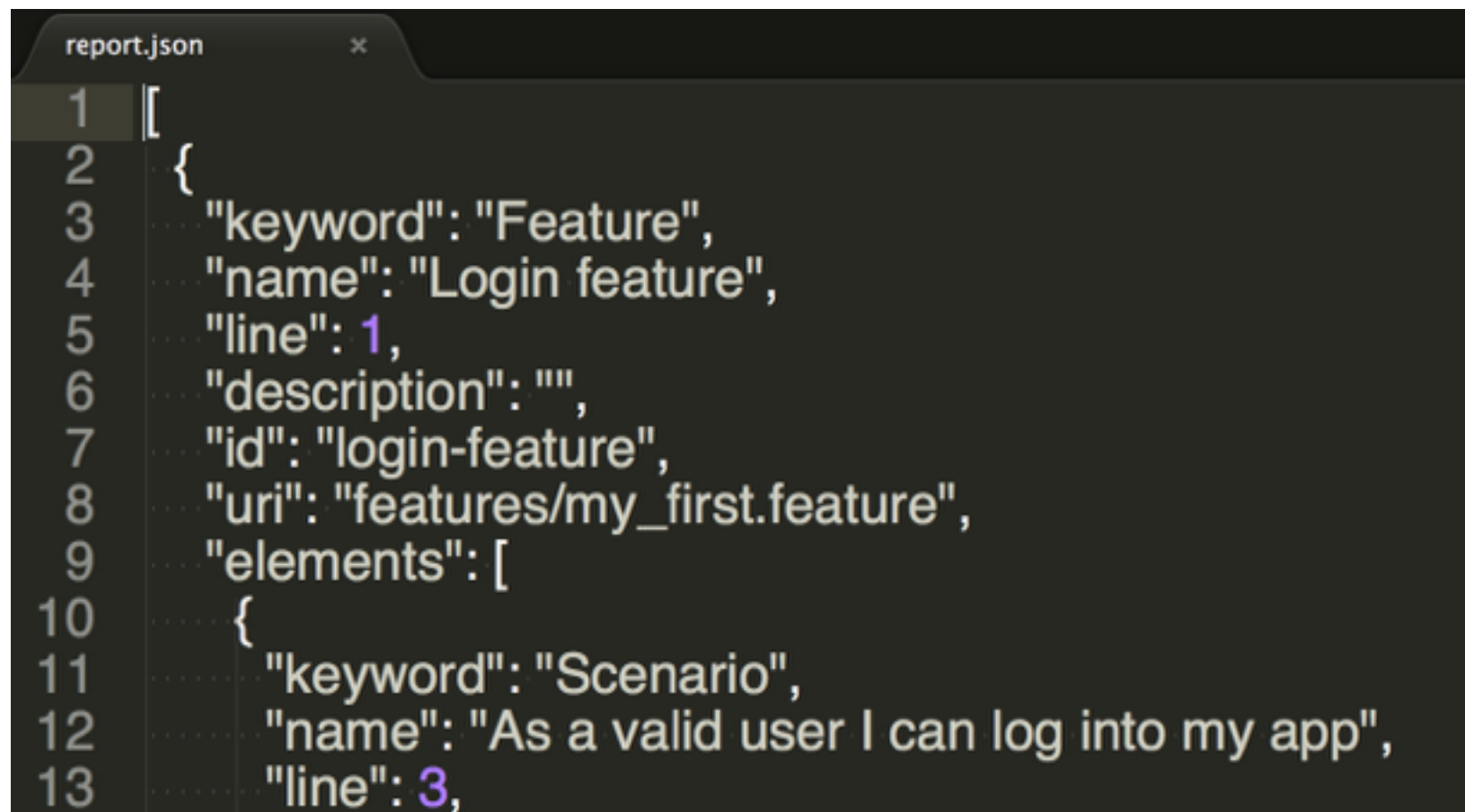
Then I see "Welcome to coolest app ever"



# Generate JSON report

\$ calabash-android run <APK file>

--format **json** --out report.json --format pretty



```
1  [
2    {
3      "keyword": "Feature",
4      "name": "Login feature",
5      "line": 1,
6      "description": "",
7      "id": "login-feature",
8      "uri": "features/my_first.feature",
9      "elements": [
10     {
11       "keyword": "Scenario",
12       "name": "As a valid user I can log into my app",
13       "line": 3,
```

# Predefined steps

- Assertion
- Input
- Button
- Gesture
- Internationalization
- Touching
- Waiting
- Screenshot
- Location

[https://github.com/calabash/calabash-android/blob/master/ruby-gem/lib/calabash-android/canned\\_steps.md](https://github.com/calabash/calabash-android/blob/master/ruby-gem/lib/calabash-android/canned_steps.md)

# Screenshot

Default is current working directory

**SCREENSHOT\_PATH=/your/path/**



# Tags

```
1 Feature: Welcome to drivebot app
2
3 @test
4 Scenario: User see all detail of application
5     Given First time to open app
6     When I see detail at next page
7     And I see detail at next page
8     And I see detail at next page
9     And I see detail at next page
10    And I see detail at next page
11    And I see detail at next page
12    And I see detail at next page
13    And I see detail at next page
14    Then I see verify your device
```

<https://github.com/cucumber/cucumber/wiki/Tags>

# Use Tags

- Feature
- Scenario
- Scenario Outline
- Example

<https://github.com/cucumber/cucumber/wiki/Tags>

# Tags

`$scalabash-android --tags @a,@b`

`$scalabash-android --tags @a --tags @b`

<https://github.com/cucumber/cucumber/wiki/Tags>

# Calling steps from Step definitions

```
Then(/^Logout from app$/) do
  steps %Q{
    And I see "Logout"
    And I press "Logout"
    And I press "lo_cfm_logout"
  }
end
```

<https://github.com/cucumber/cucumber/wiki/Calling-Steps-from-Step-Definitions>

# Scenario outline

```
2 Scenario Outline: eating
3   Given there are <start> cucumbers
4   When I eat <eat> cucumbers
5   Then I should have <left> cucumbers
6
7   Examples:
8   | start | eat | left |
9   | 12   | 5   | 7   |
10  | 20   | 5   | 15  |
```

<https://github.com/cucumber/cucumber/wiki/Scenario-Outlines>