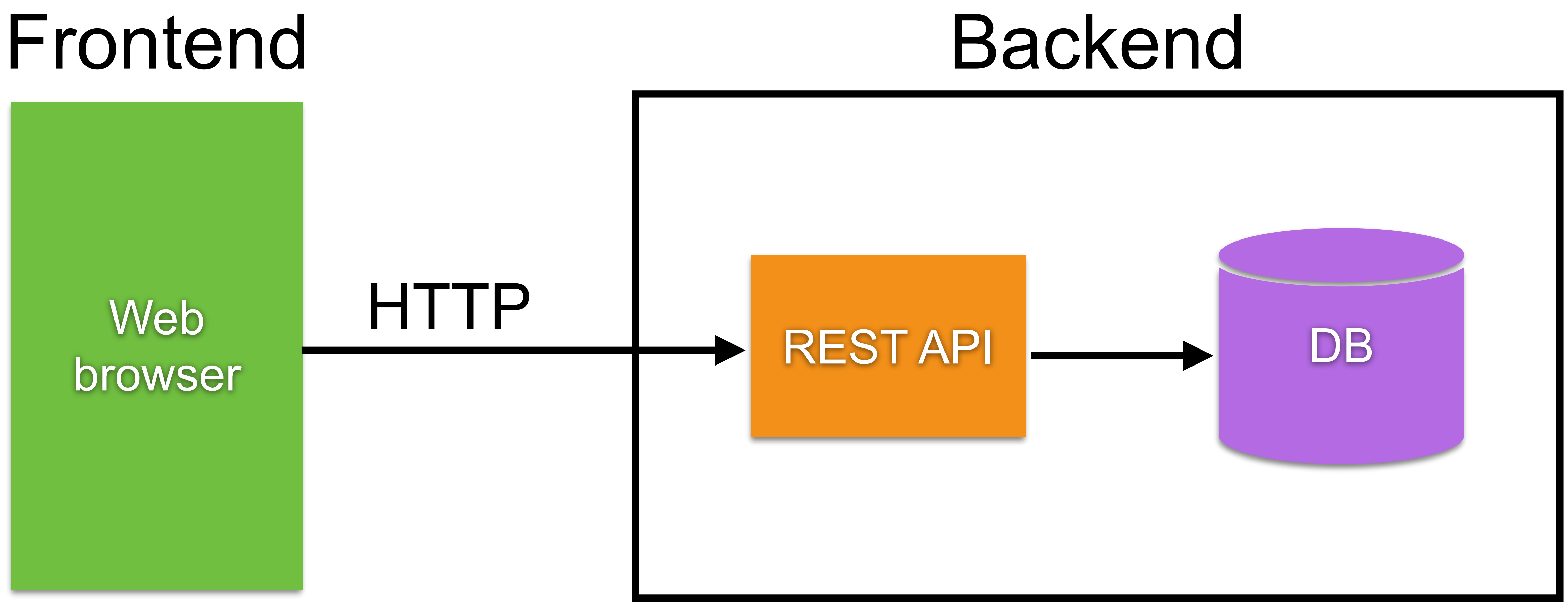
# Automated Testing with Cypress

# Automated Testing

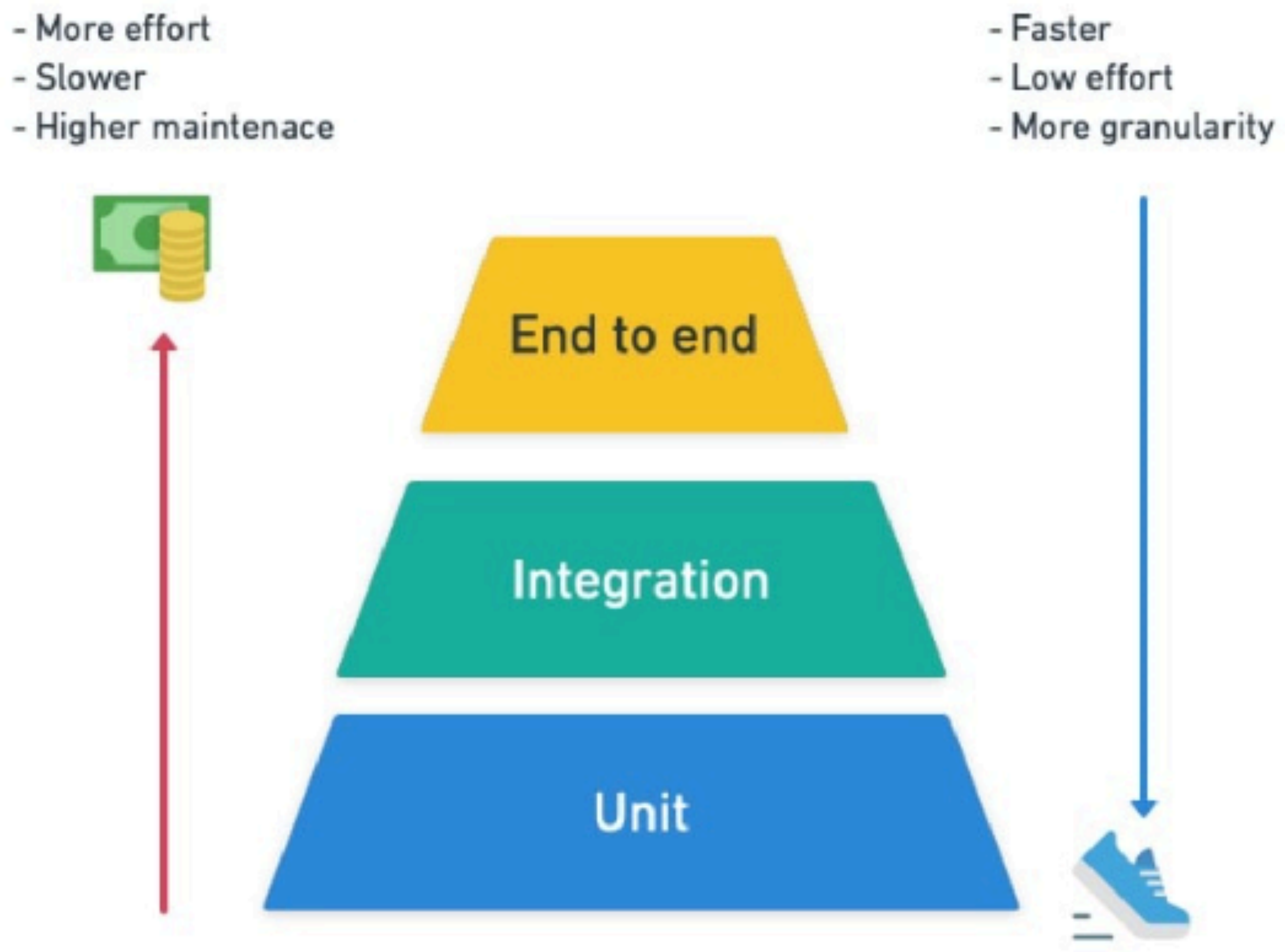
https://www.cypress.io/

# Web Application



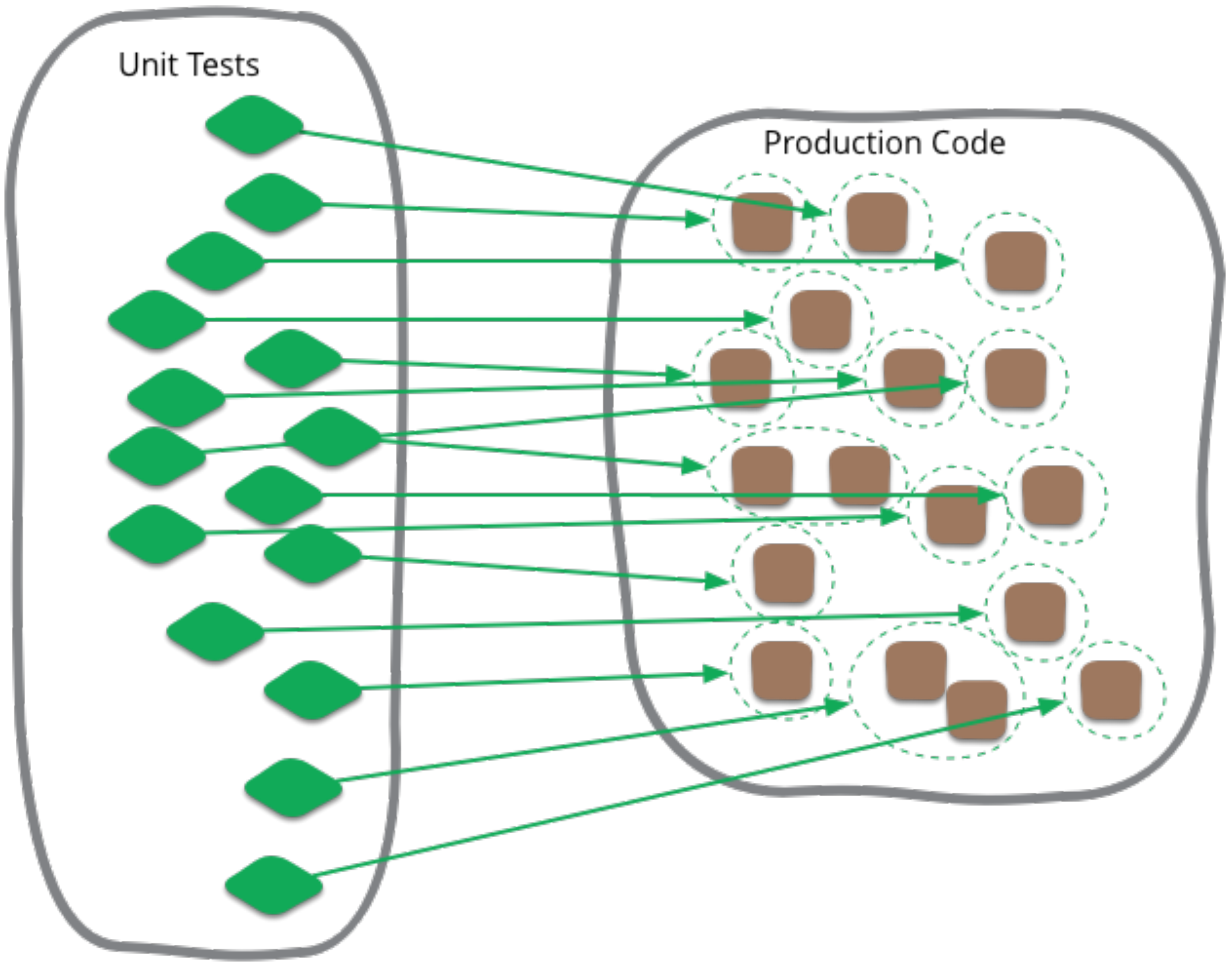
## How to test ?

# Web Application Testing



https://semaphoreci.com/blog/testing-pyramid

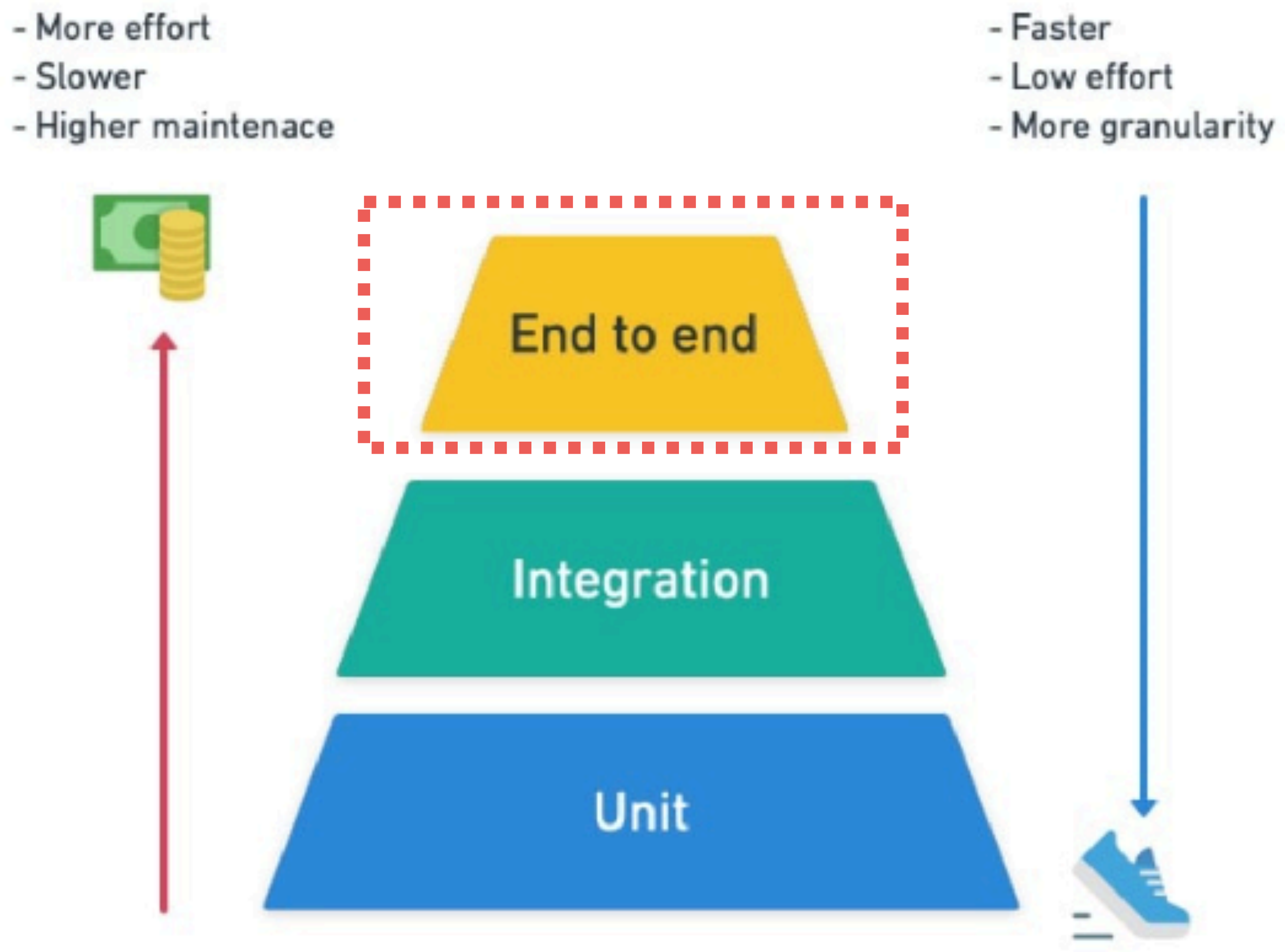
# Unit Tests



https://martinfowler.com/bliki/UnitTest.html

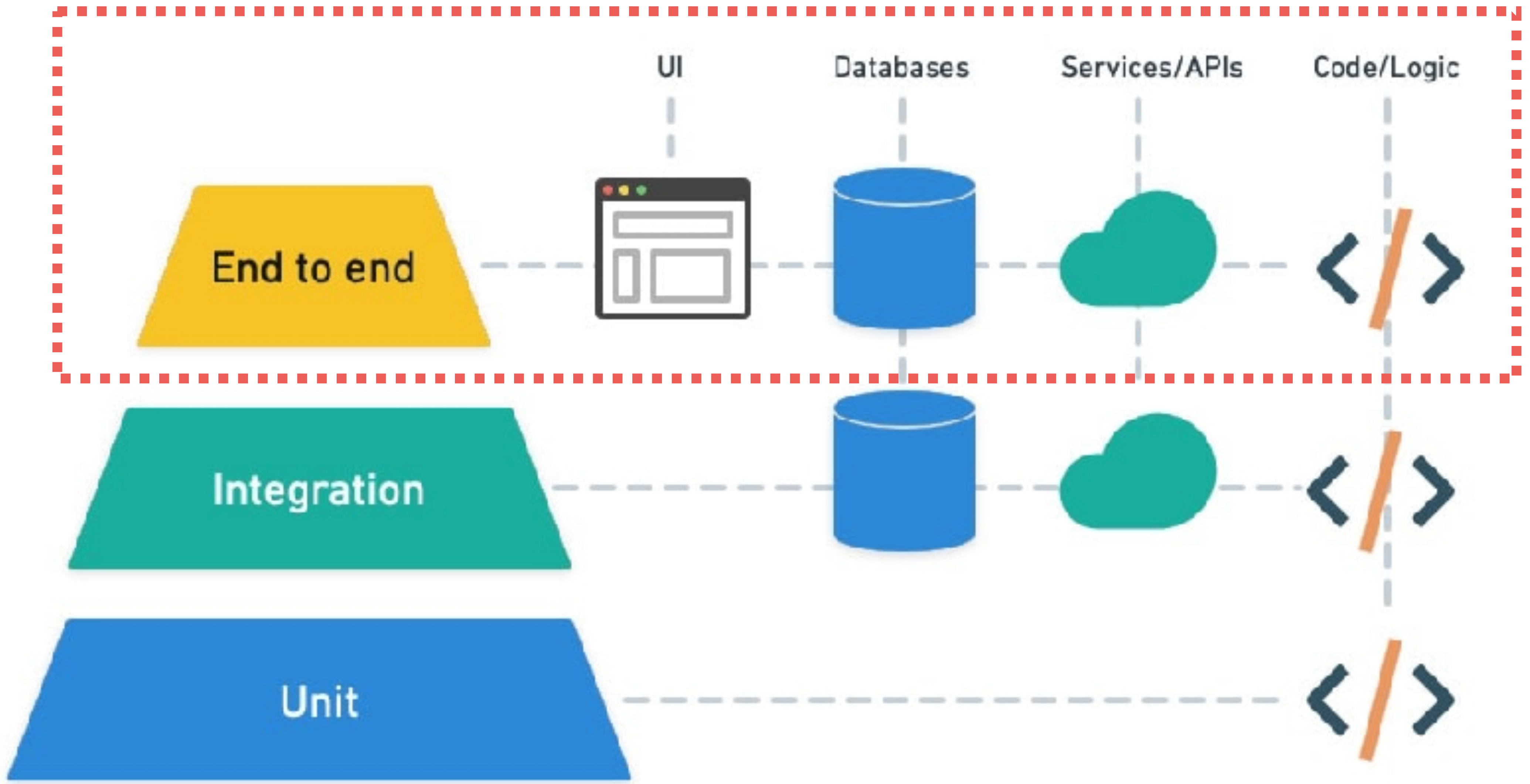
## When Unit Tests Passed …

# Web Application Testing



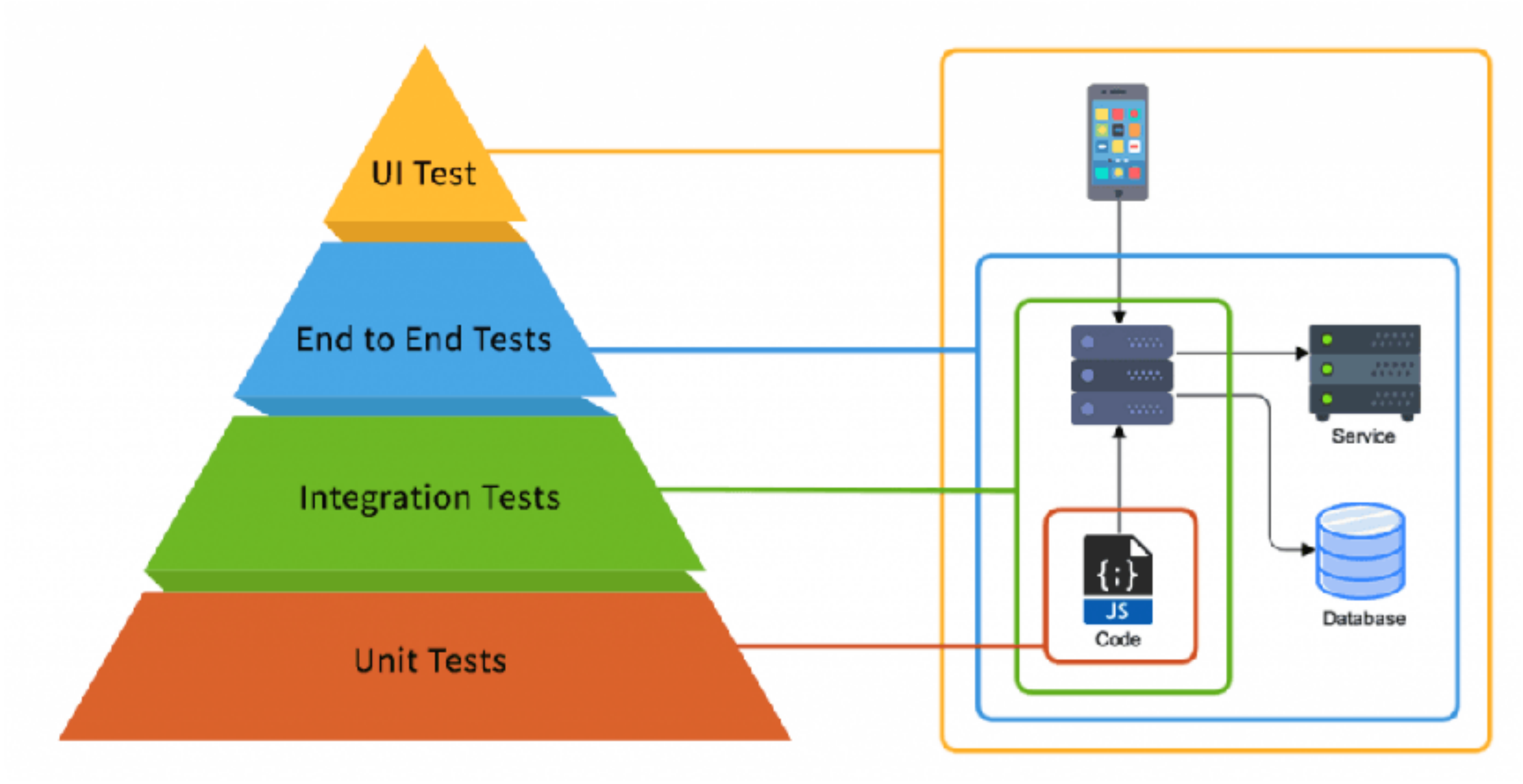
https://semaphoreci.com/blog/testing-pyramid

# Web Application Testing



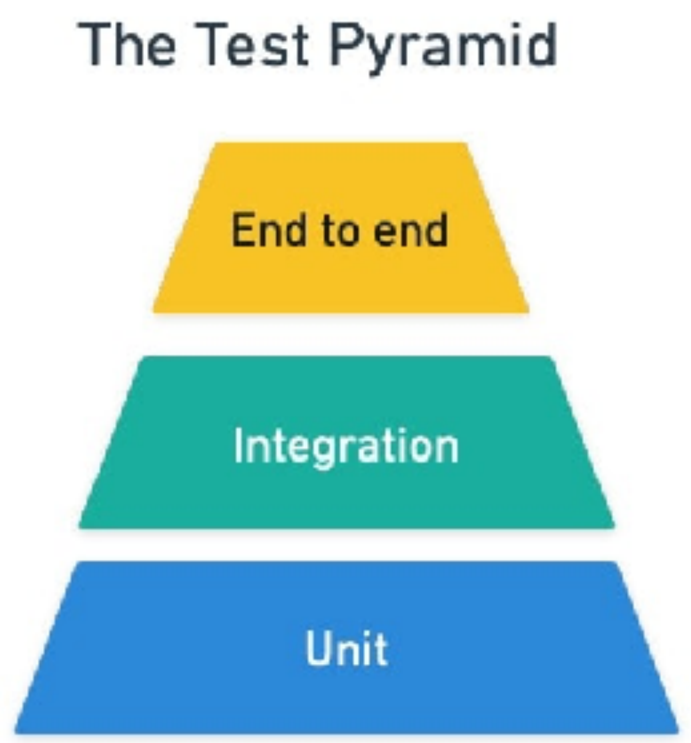
https://semaphoreci.com/blog/testing-pyramid

# Testing ?

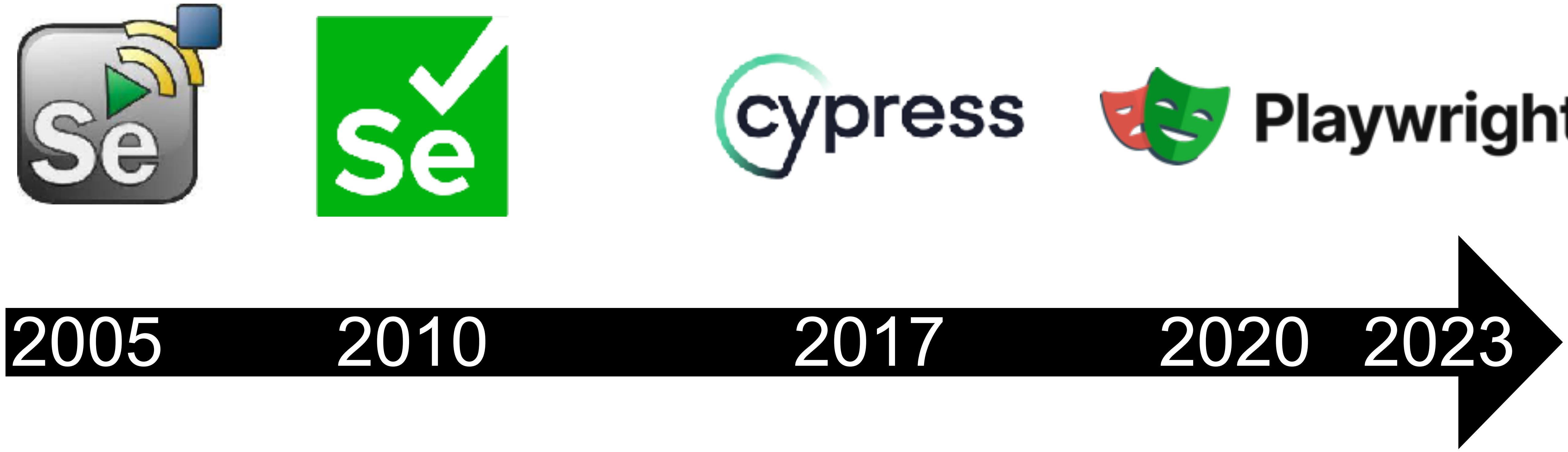

UI Test
End to End Tests
Integration Tests
Unit Tests
JS
Code
Service
Database

# Web Application Testing

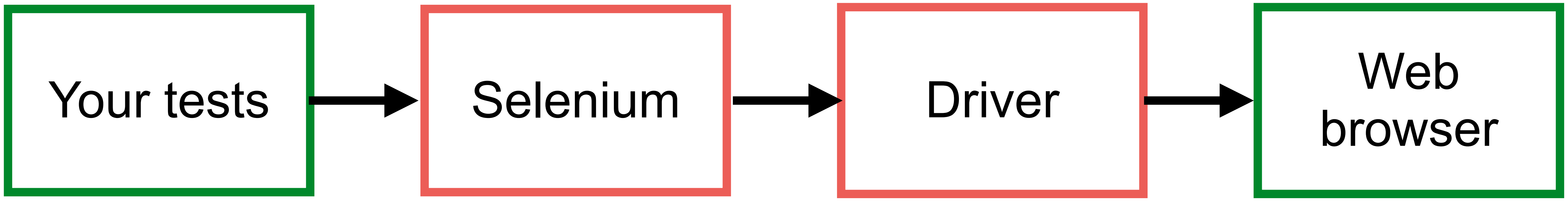
Run on real browser
Load read app
Interact with app like a user



# History



# Run on real browser !!

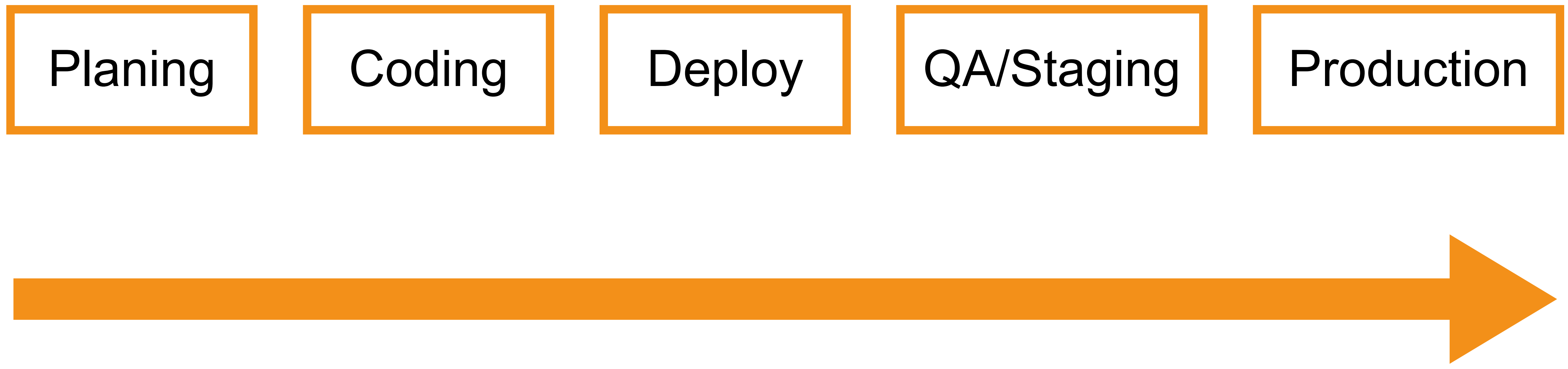


*Slow, flaky, unreliable tests*

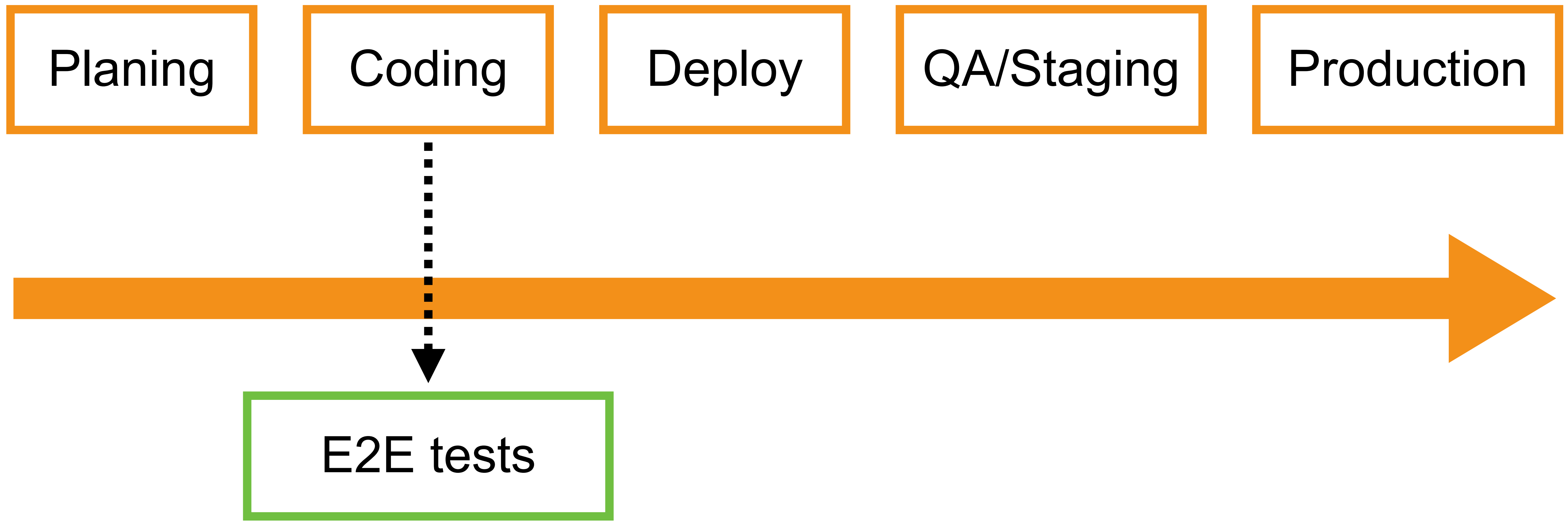
## End-to-End Test

Easy to install
Easy to write tests
Simple to run
Easy to debug when a test failed

## When should write E2E test ?



# When should write E2E tests ?



# Cypress

Focus on **End-to-End test**

Support modern web browsers

Works on any frontend framework and website

Tests are written in JavaScript

For developer and QA/Tester

Easy

Fast

# Before Cypress

Mocha | Jasmine | Karma | QUnit

Chai | Expect.js

Protractor | Nightwatch | WebDriver | Selenium

Sinon | Jest

# Cypress

**All-in-one testing framework**

Assertion library

Mocking, Stubbing

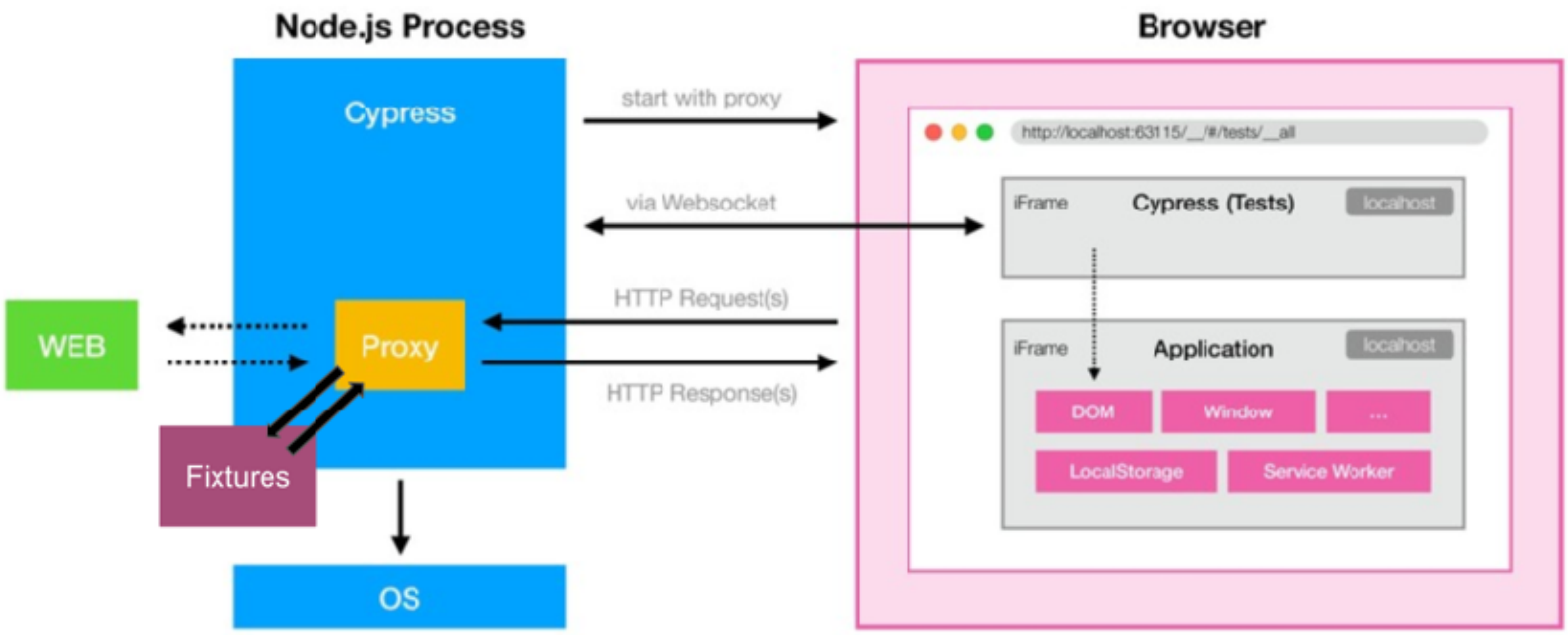
Not use selenium

Mocha | Chai | Sinon

Minimatch | Lodash | jQuery | Bluebird

https://docs.cypress.io/guides/references/bundled-libraries

# Architecture



https://www.codecentric.de/wissens-hub/blog/cypress-ui-end2end-testing

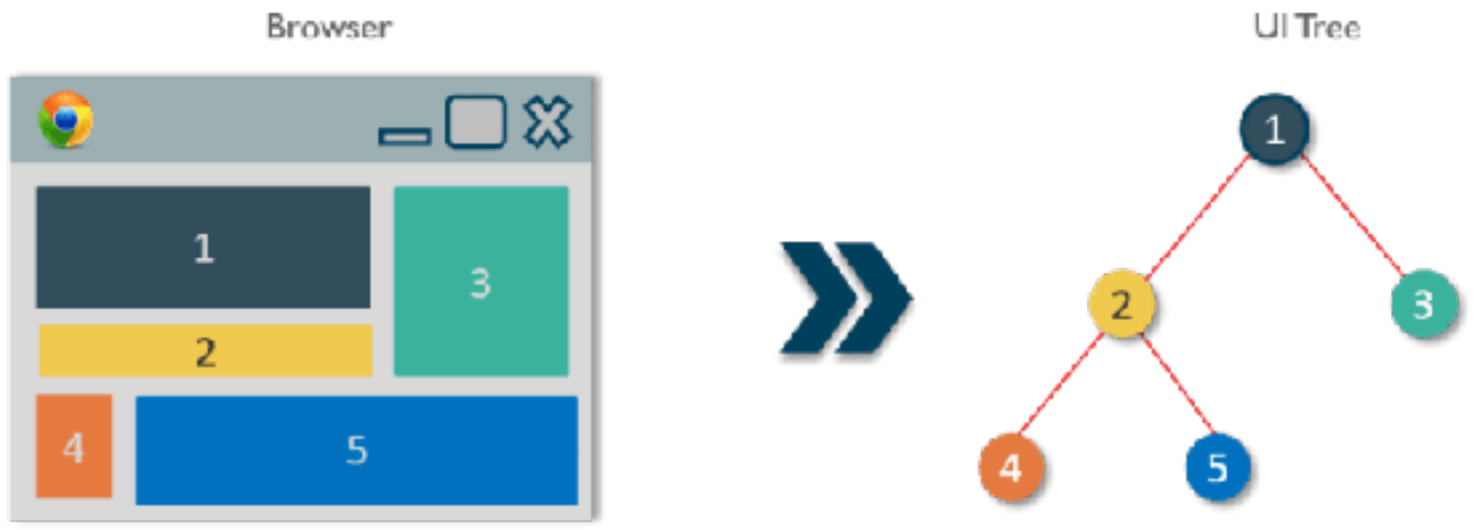
# Cypress Testing

| | |
|---|---|
| E2E test Web Browser | Component Test |
| API Test | Snapshot Test |

https://docs.cypress.io/guides/core-concepts/testing-types

# Component Testing

Test in each component in app
Fast and reliable
Not call external APIs or services
Test logic and display in each component



https://docs.cypress.io/guides/references/bundled-libraries

# Features

Time travel

Real time reload

Stub, spy and clock

Debuggability

Automatic waiting

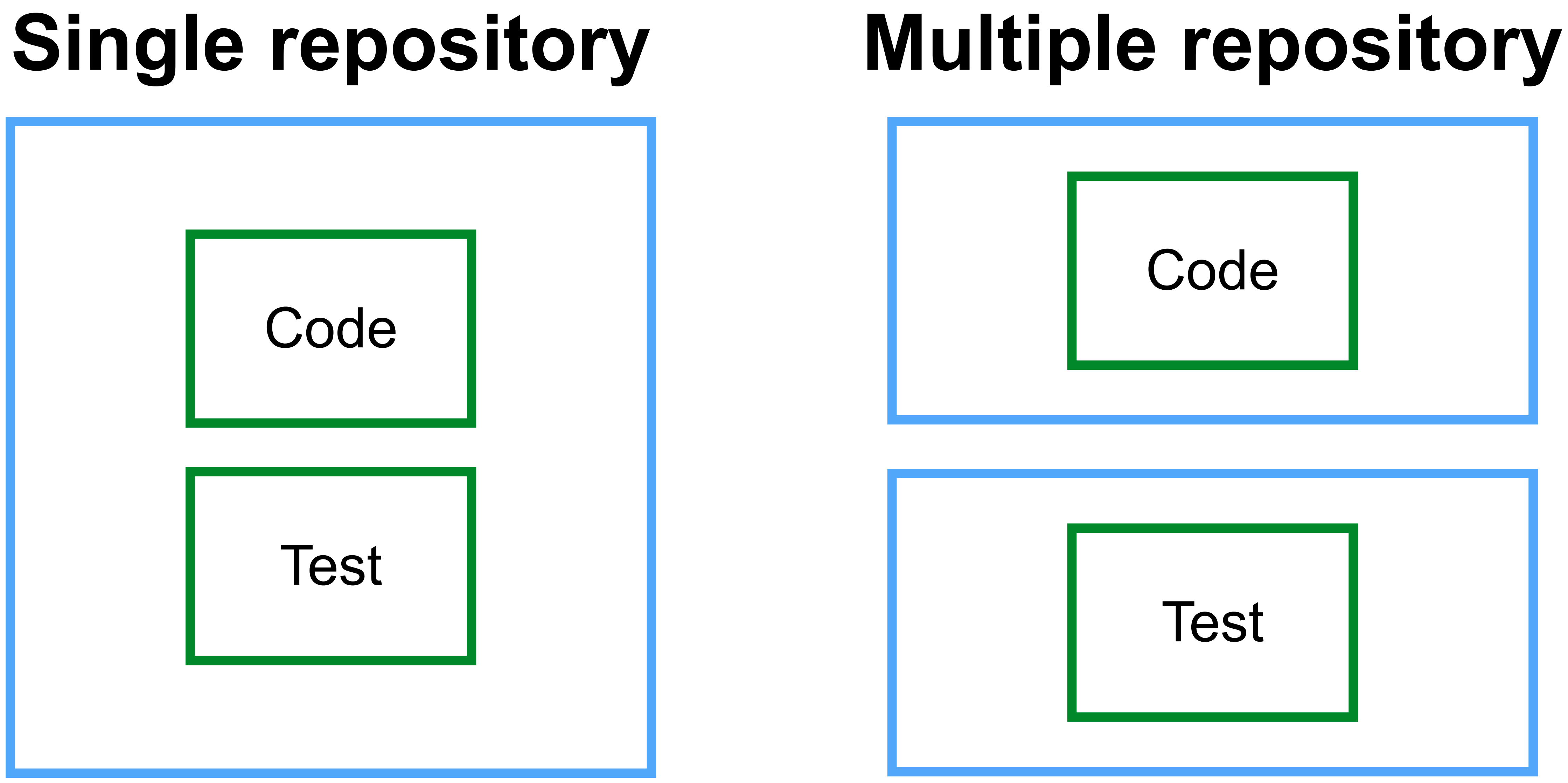
Screenshot and video

# Learning paths

1. Setup
2. Write tests
3. Run tests
4. Debugging
5. Repoting
6. Recording

## Repository to keep tests ?

# Repository to keep tests ?



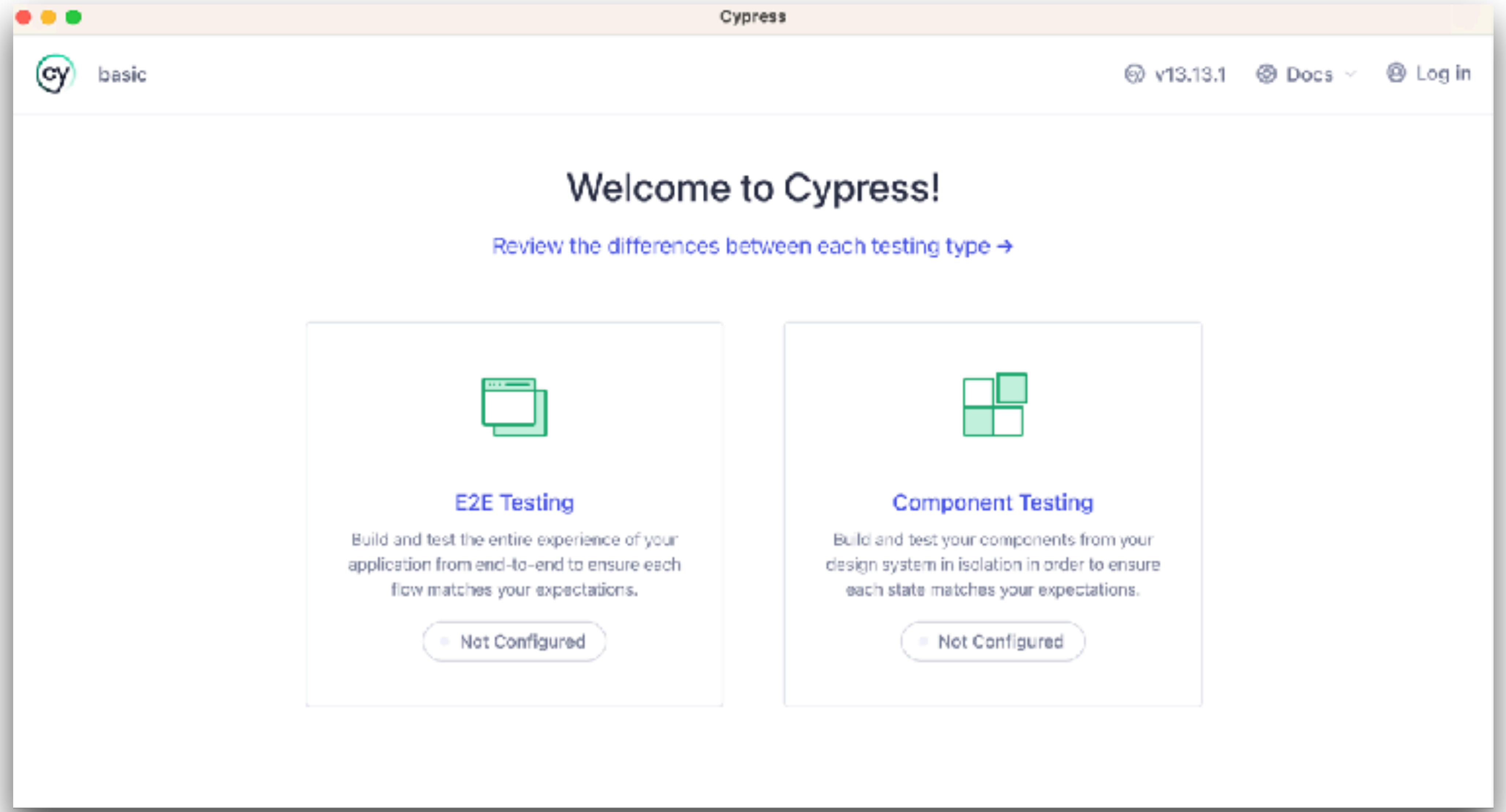
## Setup for Cypress

## Start with Cypress

```
$npm init -y
$npm i -D cypress
$npx cypress open
```

## Cypress UI

$npx cypress open



Welcome to Cypress!

Review the differences between each testing type →

E2E Testing

Build and test the entire experience of your application from end-to-end to ensure each flow matches your expectations.

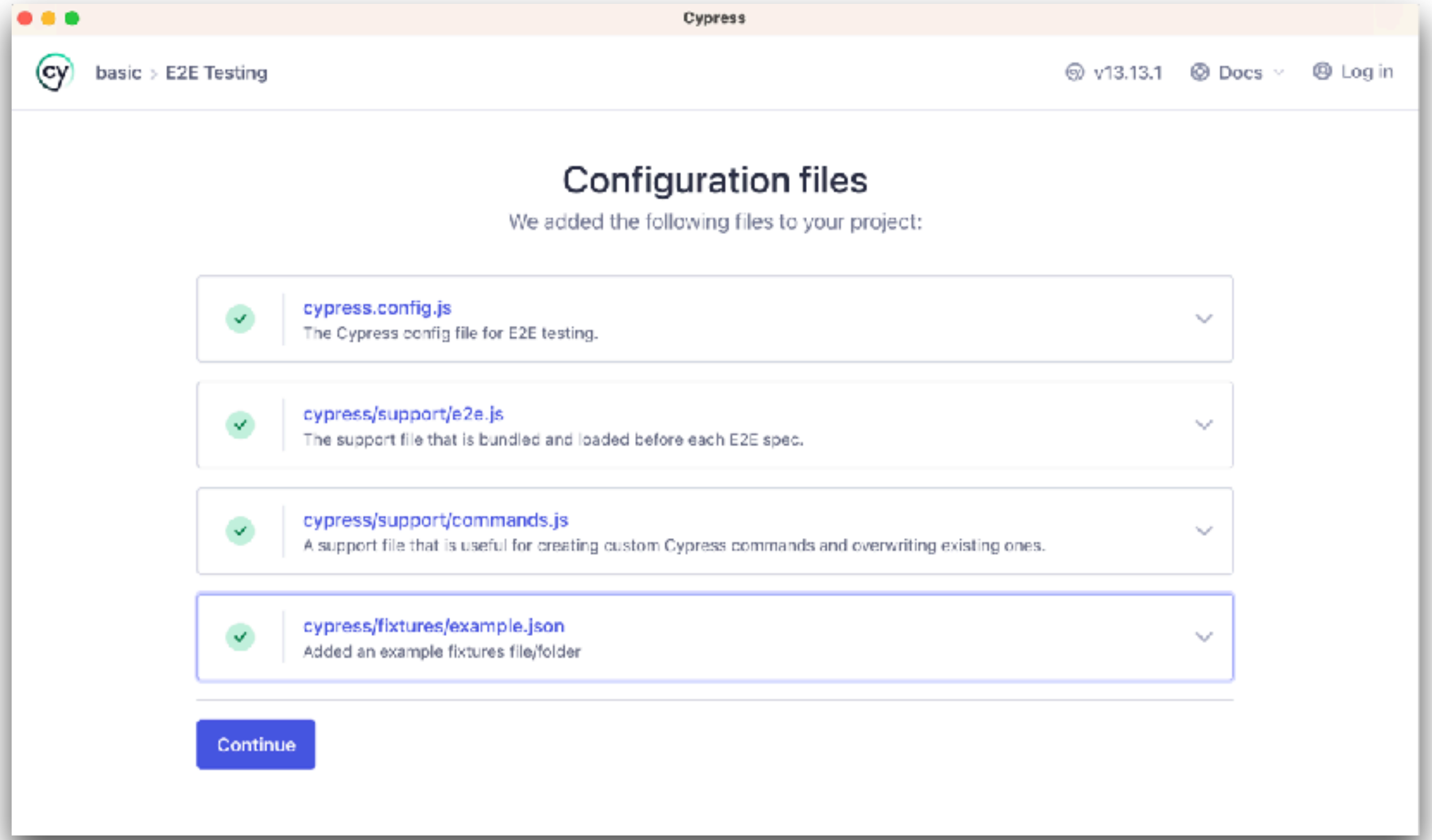
Not Configured

Component Testing

Build and test your components from your design system in isolation in order to ensure each state matches your expectations.

Not Configured

# Structure of cypress project



Configuration files

We added the following files to your project:

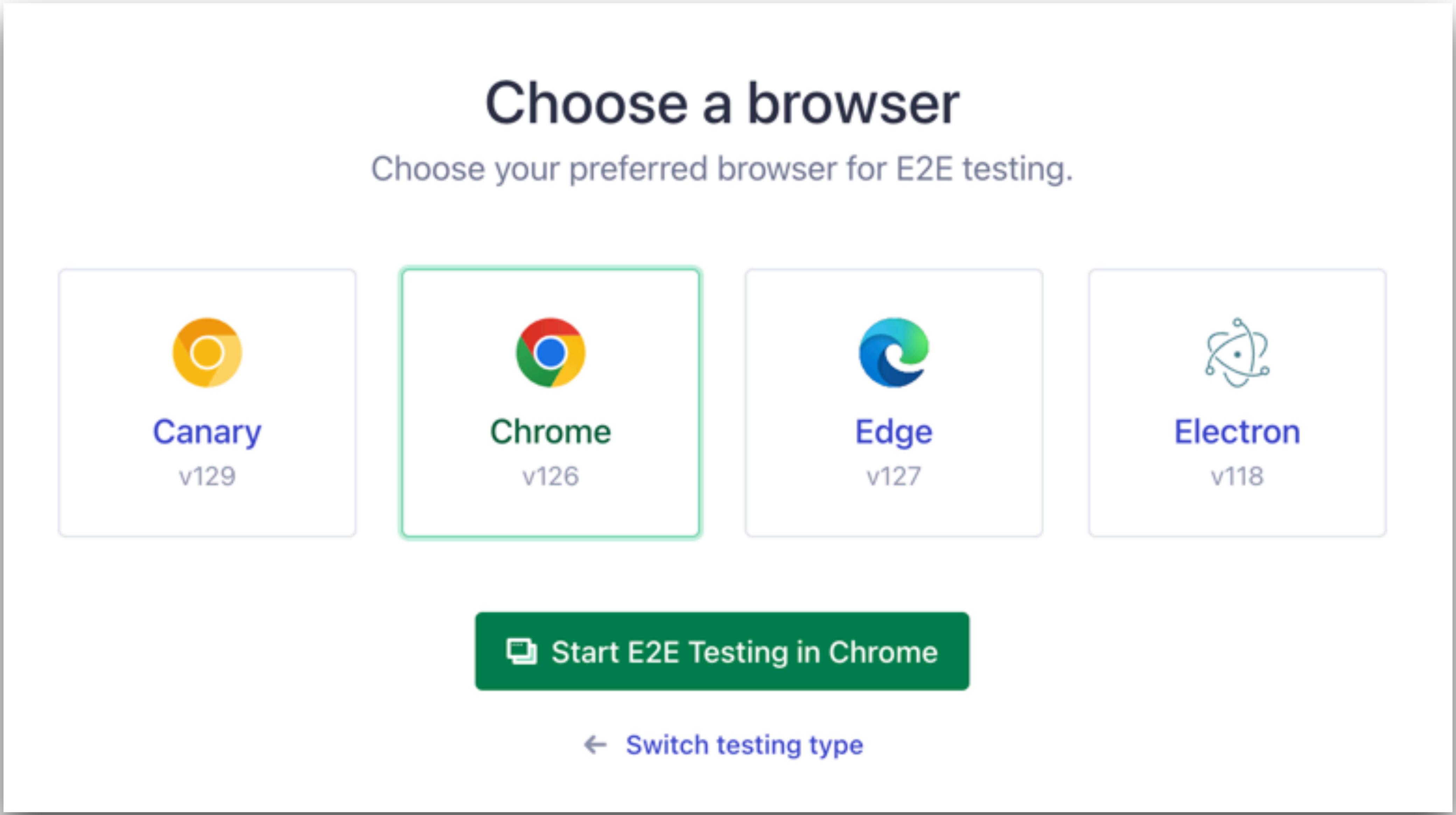
cypress.config.js
The Cypress config file for E2E testing.

cypress/support/e2e.js
The support file that is bundled and loaded before each E2E spec.

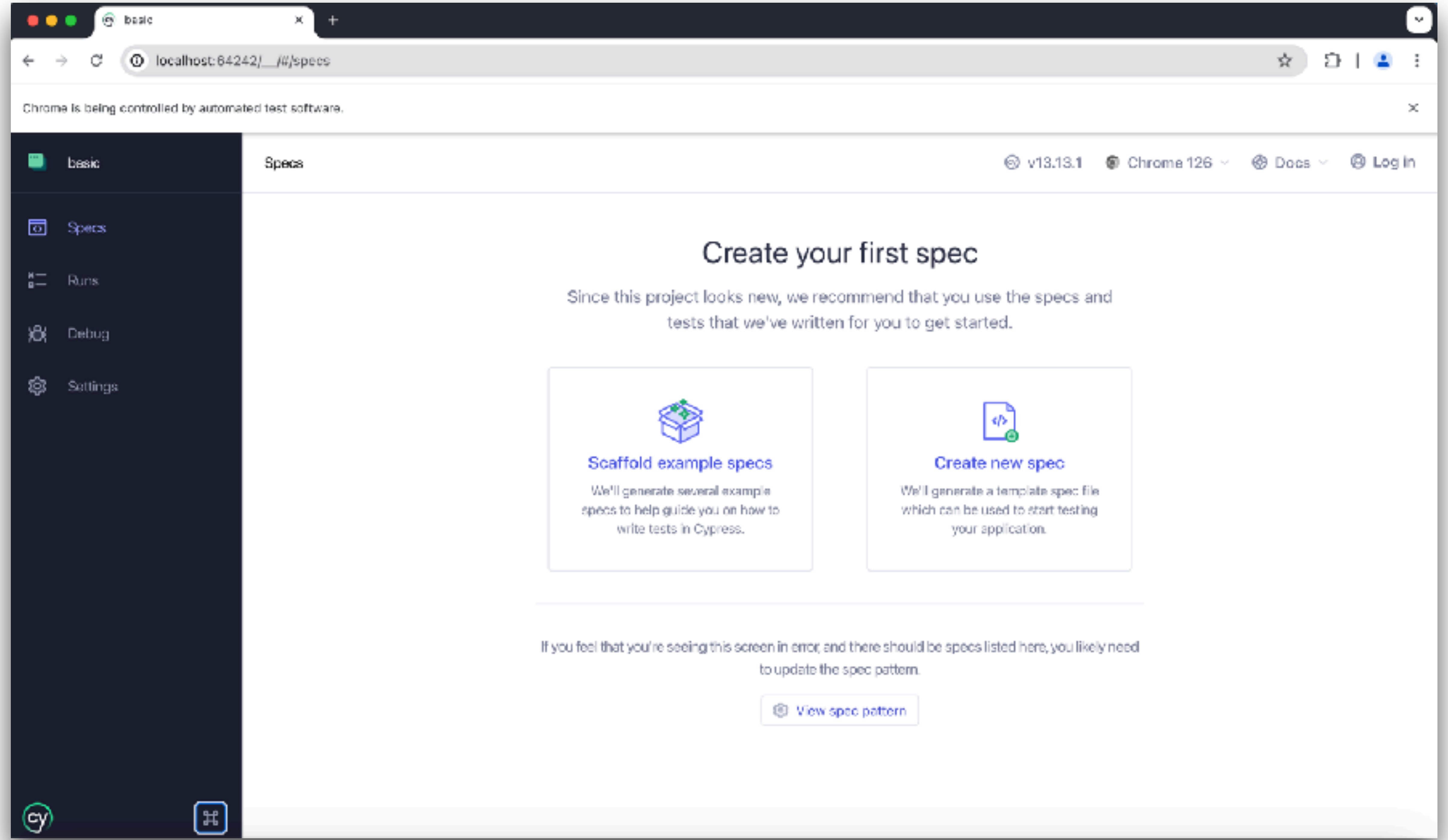
cypress/support/commands.js
A support file that is useful for creating custom Cypress commands and overwriting existing ones.

cypress/fixtures/example.json
Added an example fixtures file/folder

# Web browsers for E2E test



# Create a first tests/specs



basic

localhost:64242/__/#/specs

Chrome is being controlled by automated test software.

Create your first spec

Since this project looks new, we recommend that you use the specs and tests that we've written for you to get started.

**Scaffold example specs**

We'll generate several example specs to help guide you on how to write tests in Cypress.

**Create new spec**

We'll generate a template spec file which can be used to start testing your application.

If you feel that you're seeing this screen in error, and there should be specs listed here, you likely need to update the spec pattern.

View spec pattern

# Folder structure of cypress

| Folder name | Description |
|---|---|
| **e2e** | Keep specs/test files (.js, .ts) |
| **fixtures** | Keep static data that can be used in tests |
| **supports** | Config files for e2e and component test, custom commands |
| downloads | Keep download files in tests |
| screenshot | Test results |
| video | Test results |

https://docs.cypress.io/guides/references/configuration#Folders--Files

## Configuration file

cypress.config.js

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {
    setupNodeEvents(on, config) {
      // implement node event listeners here
    },
  },
});
```

https://docs.cypress.io/guides/references/configuration

## Write a test

Create file in folder **/cypress/e2e**

## Good Test ?

# Good Test ?

## F.I.R.S.T and U

**F**ast
**I**ndependent / **I**solate
**R**epeatable
**S**elf-verify
**T**imely / **T**horough
**U**nderstandable

# Hello Google

## Create file /e2e/google.cy.js

```
/// <reference types="cypress" />

describe("Search with google.com", () => {

  it("Success with search by keyword=cypress", () => {

    // Arrange
    cy.visit("https://www.google.com");

    // Act
    cy.get('[name="q"]').type("cypress");
    cy.get('[name="q"]').type("{enter}");

    // Assert
    cy.get("#result-stats").should("be.visible");
    cy.get("#result-stats").contains("ผลการค้นหาประมาณ");
    cy.get("#result-stats").should("include.text", "ผลการค้นหาประมาณ");
  });

});
```

# Commands

```
/// <reference types="cypress" />

describe("Search with google.com", () => {

  it("Success with search by keyword=cypress", () => {

   // Act
    cy.get('[name="q"]').type("cypress");
    cy.get('[name="q"]').type("{enter}");

    // Assert
    cy.get(“#result-stats").should("be.visible");
    cy.get("#result-stats").contains("ผลการค้นหาประมาณ");
  });

});
```

## Commands with timeout !!

```
/// <reference types="cypress" />

describe("Search with google.com", () => {

  it("Success with search by keyword=cypress", () => {

   // Act
    cy.get('[name="q"]').type("cypress");
    cy.get('[name="q"]').type("{enter}");

    // Assert

    cy.get("#result-stats", { timeout: 5000})
          .should("be.visible");

    cy.get("#result-stats").contains("ผลการค้นหาประมาณ");
  });

});
```

# Assertion

```
/// <reference types="cypress" />

describe("Search with google.com", () => {

  it("Success with search by keyword=cypress", () => {

   // Act
    cy.get('[name="q"]').type("cypress");
    cy.get('[name="q"]').type("{enter}");

    // Assert
    cy.get("#result-stats").should("be.visible");
    cy.get("#result-stats").contains("ผลการค้นหาประมาณ");
  });

});
```

# Web element locator (Selector)

```
<button
  id="main"
  class="btn btn-large"
  name="submission"
  role="button"

  data-cy="submit"
  data-testid="submit"

>
  Submit
</button>
```

https://docs.cypress.io/api/commands/get#Selector

# Web element locator (Selector)

| Name | Description |
|---|---|
| id | cy.get('#main').click() |
| name | cy.get('[name="submission"]').click() |
| class | cy.get('.btn.btn-large').click() |
| xPath | Not support |
| **data-cy** | **cy.get('[data-cy="submit"]').click()** |

https://docs.cypress.io/guides/references/best-practices#Selecting-Elements

## Run test

# Run test in command-line

**$npm test**

$npx cypress run

File package.json

```
{
  "version": "1.0.0",
  "main": "index.js",

  "scripts": {
    "test": "cypress run"
  },

}
```

***Headless mode by default***

# Run test in command-line

| Browser | Run in command-line |
|---|---|
| Electron | cypress run<br>cypress run --headed |
| Google Chrome | cypress run --browser chrome<br>cypress run --browser chrome --headed |
| Microsoft Edge | cypress run --browser edge<br>cypress run --browser edge --headed |
| Firefox | cypress run --browser firefox<br>cypress run --browser firefox --headed |

# Test report

Spec

JUnit

Teamcity

Custom report

https://docs.cypress.io/guides/tooling/reporters

# Mocha Spec Report

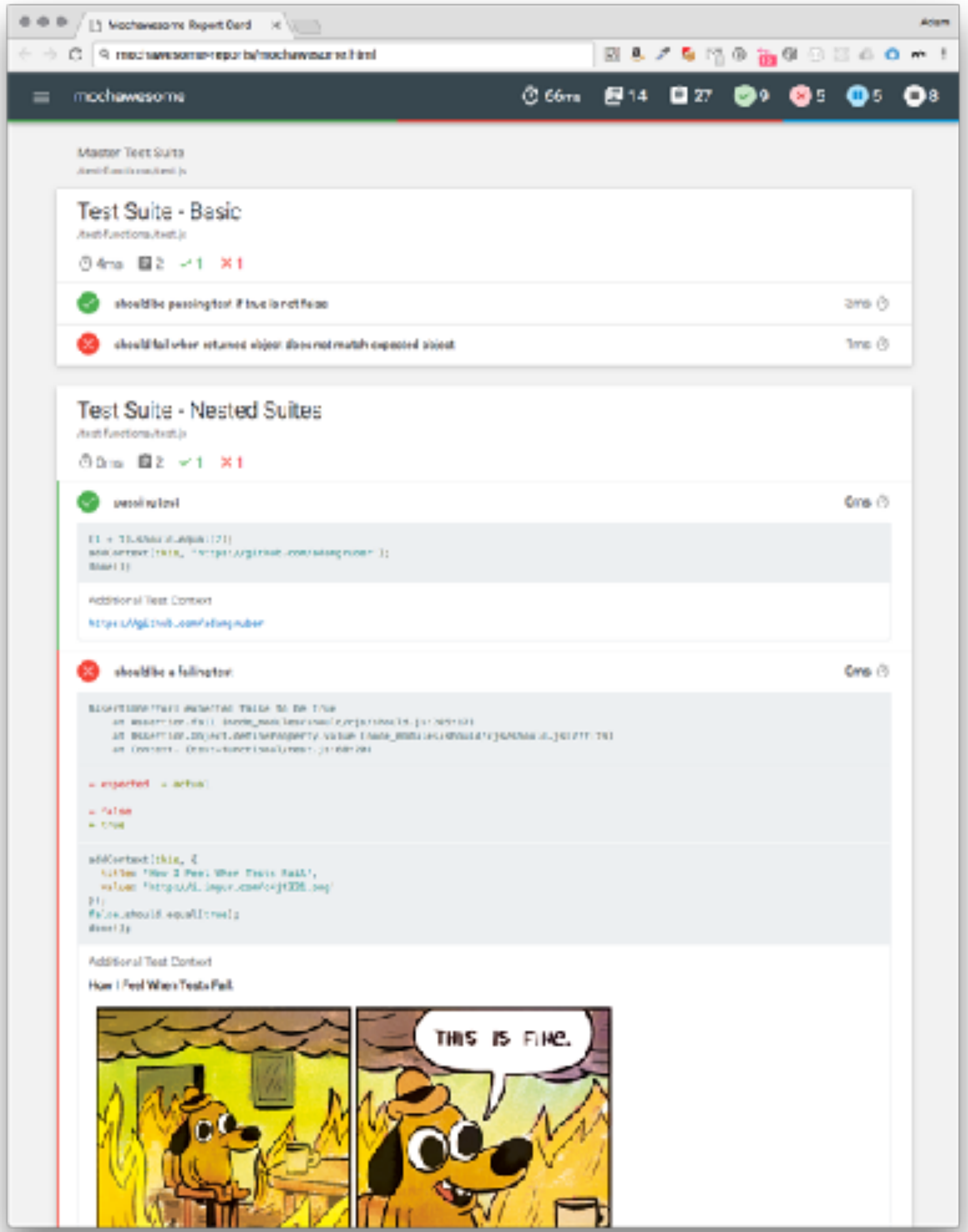
$npx cypress run **--reporter spec** -b chrome

$npx cypress run **--reporter dot** -b chrome

$npx cypress run **--reporter dot** -b chrome

$npx cypress run **--reporter junit** -b chrome

# Mocha Awesome



mochawesome 66ms 14 27 9 5 5 8

Master Test Suite

Test Suite - Basic

./test/functions/test.js

4ms 2 ✓1 ✕1

should be passing test if true is not false 2ms

should fail when returned object does not match expected object 1ms

Test Suite - Nested Suites

./test/functions/test.js

0ms 2 ✓1 ✕1

passing test 0ms

```
[illegible]
```

Additional Test Context

[illegible]

should be a failing test 0ms

```
[illegible]
```

Additional Test Context

How I Feel When Tests Fail

https://github.com/adamgruber/mochawesome

## Install

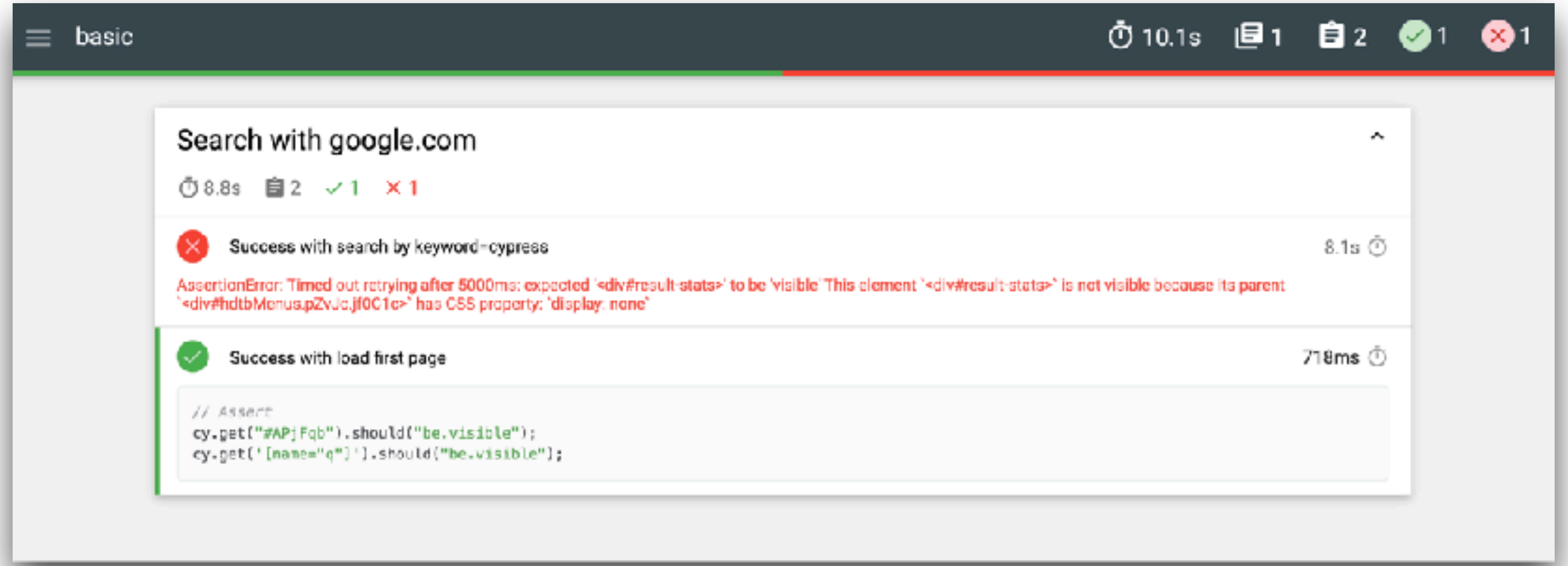
```
$npm i -D mochawesome
$npm i -D mochawesome-merge
$npm i -D mochawesome-report-generator
```

# Config cypress.config.js

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({

  reporter: "mochawesome",
  reporterOptions: {
    reportDir: 'cypress/results',
    overwrite: false,
    html: true,
    json: true,
  },

  video: true,
  screenshotsFolder: "cypress/screenshots",
  execTimeout: 6000,

});
```

# Run test

$npx cypress run -b chrome



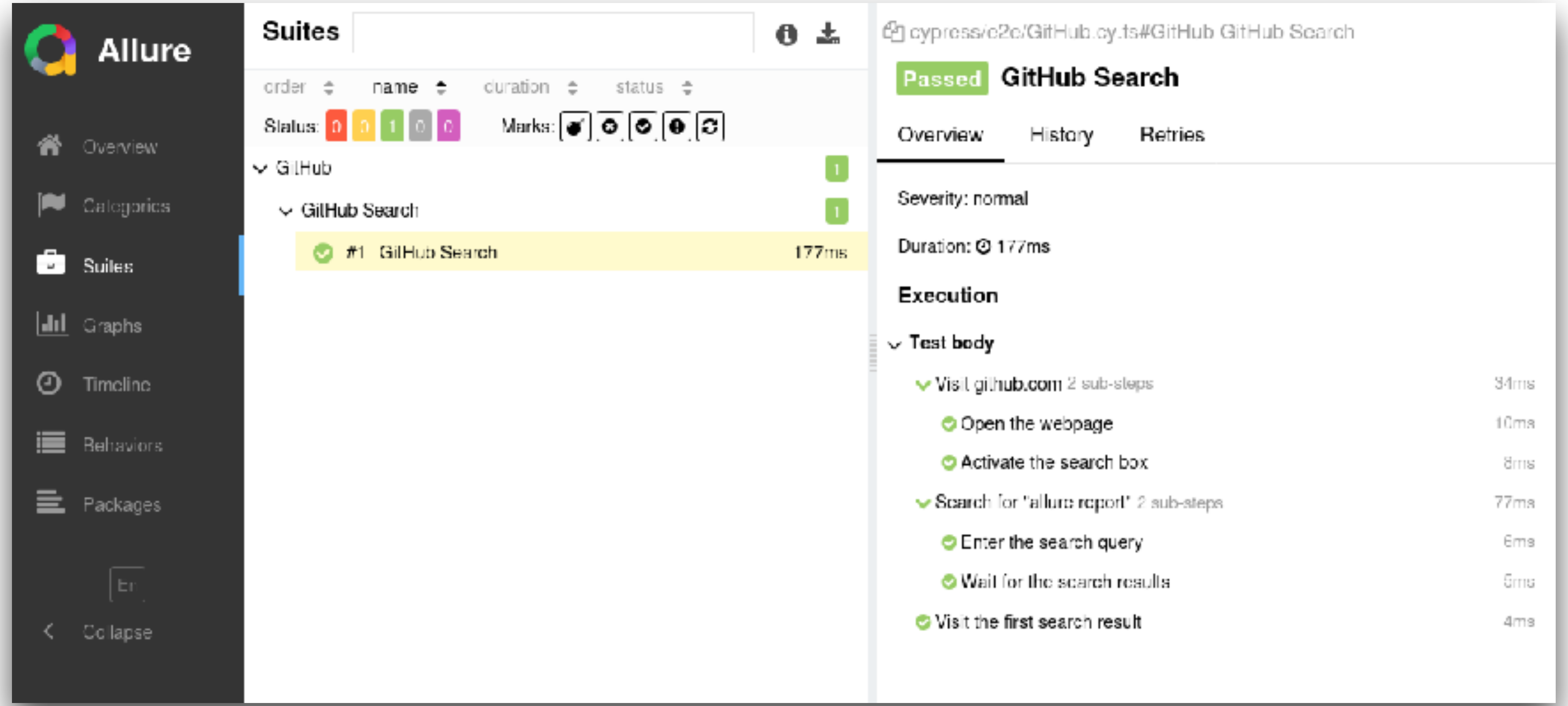
Success with search by keyword-cypress 8.1s

```
AssertionError: Timed out retrying after 5000ms: expected '<div#result-stats>' to be 'visible' This element '<div#result-stats>' is not visible because its parent
'<div#hdtbMenus.pZvJc.jf0C1c>' has CSS property: 'display: none'
```

Success with load first page 718ms

```
// Assert
cy.get("#APjFqb").should("be.visible");
cy.get('[name="q"]').should("be.visible");
```

# Allure Cypress



GitHub 1
GitHub Search 1
#1 GitHub Search 177ms

cypress/e2e/GitHub.cy.ts#GitHub GitHub Search

Passed GitHub Search

Severity: normal

Duration: 177ms

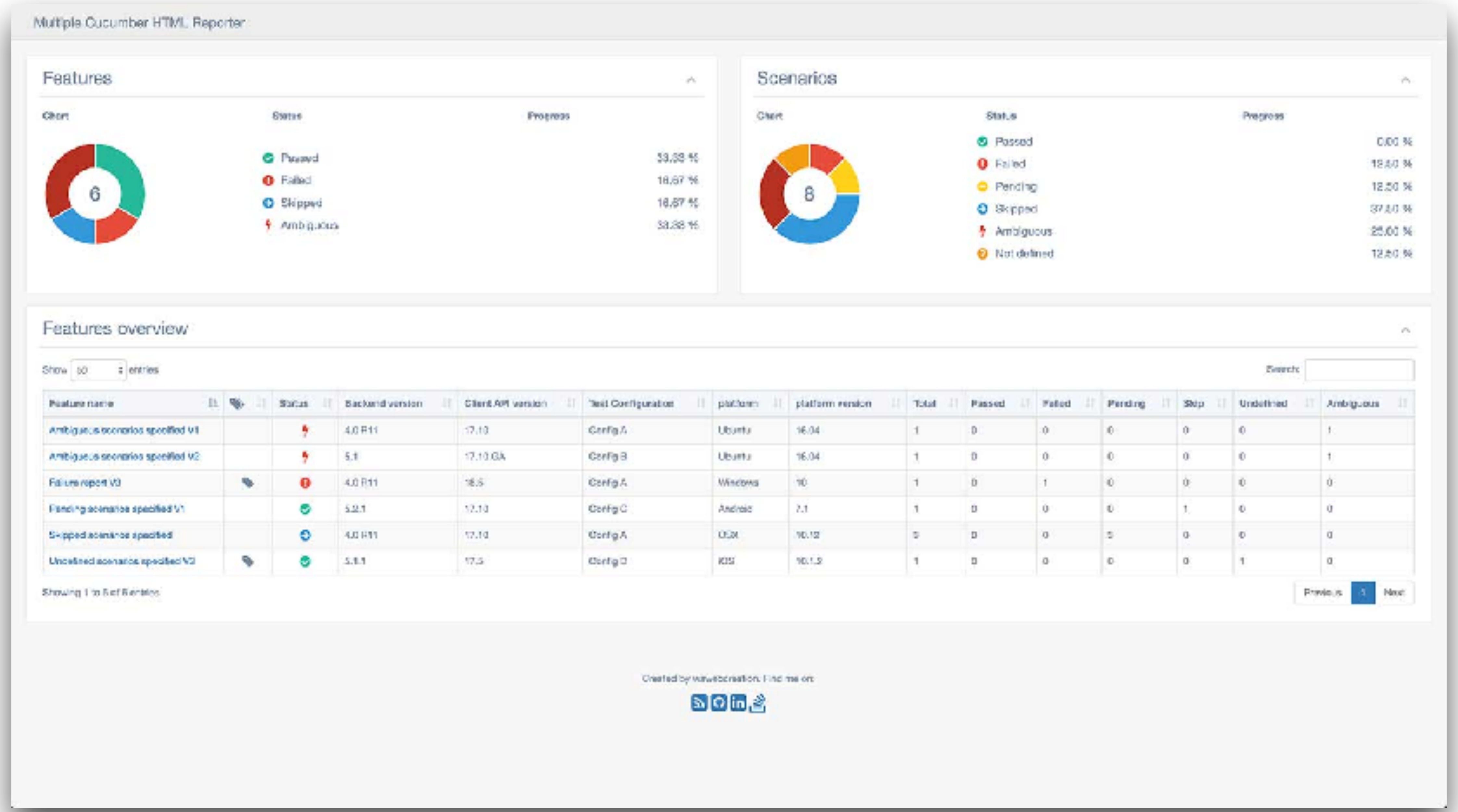
Execution

Test body

```
Visit github.com 2 sub-steps                       34ms
    Open the webpage                               10ms
    Activate the search box                         8ms
Search for "allure report" 2 sub-steps             77ms
    Enter the search query                          6ms
    Wait for the search results                     5ms
Visit the first search result                       4ms
```

https://allurereport.org/docs/cypress/

# Cucumber HTML Report



Multiple Cucumber HTML Reporter

## Features

| Chart | Status | Progress |
|---|---|---|
| 6 | ✓ Passed | 33.33 % |
| | ❗ Failed | 16.67 % |
| | ➜ Skipped | 16.67 % |
| | ⚡ Ambiguous | 33.33 % |

## Scenarios

| Chart | Status | Progress |
|---|---|---|
| 8 | ✓ Passed | 0.00 % |
| | ❗ Failed | 12.50 % |
| | ● Pending | 12.50 % |
| | ➜ Skipped | 37.50 % |
| | ⚡ Ambiguous | 25.00 % |
| | ? Not defined | 12.50 % |

## Features overview

Show 50 entries  Search:

| Feature name | 🏷 | Status | Backend version | Client API version | Test Configuration | platform | platform version | Total | Passed | Failed | Pending | Skip | Undefined | Ambiguous |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ambiguous scenarios specified V1 | | ⚡ | 4.0 R11 | 17.10 | Config A | Ubuntu | 16.04 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Ambiguous scenarios specified V2 | | ⚡ | 5.1 | 17.10 GA | Config B | Ubuntu | 16.04 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| Failure report V3 | 🏷 | ❗ | 4.0 R11 | 16.5 | Config A | Windows | 10 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Pending scenarios specified V1 | | ✓ | 5.2.1 | 17.10 | Config C | Android | 7.1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| Skipped scenarios specified | | ➜ | 4.0 R11 | 17.10 | Config A | OSX | 10.12 | 5 | 0 | 0 | 5 | 0 | 0 | 0 |
| Undefined scenarios specified V2 | 🏷 | ✓ | 5.1.1 | 17.5 | Config D | iOS | 10.1.2 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

Showing 1 to 6 of 6 entries  Previous 1 Next

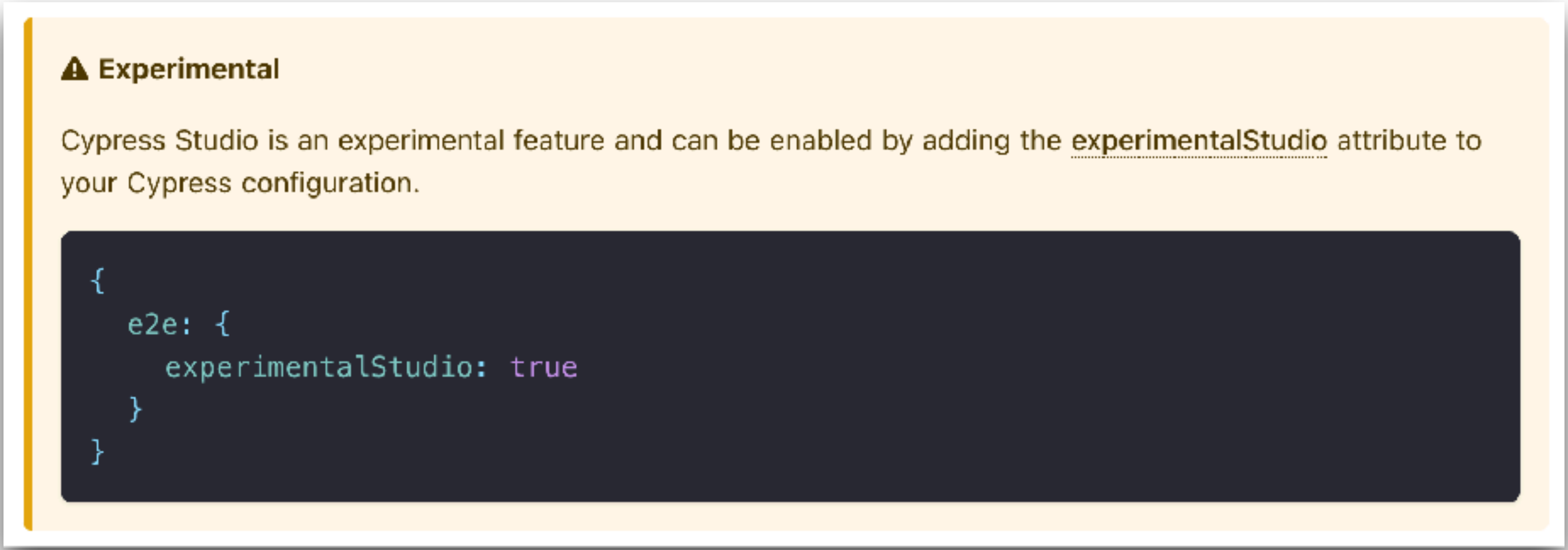
Created by wswebcreation. Find me on:

https://github.com/WasiqB/multiple-cucumber-html-reporter

## Cypress recording ?

# Cypress Studio

## Experimental feature



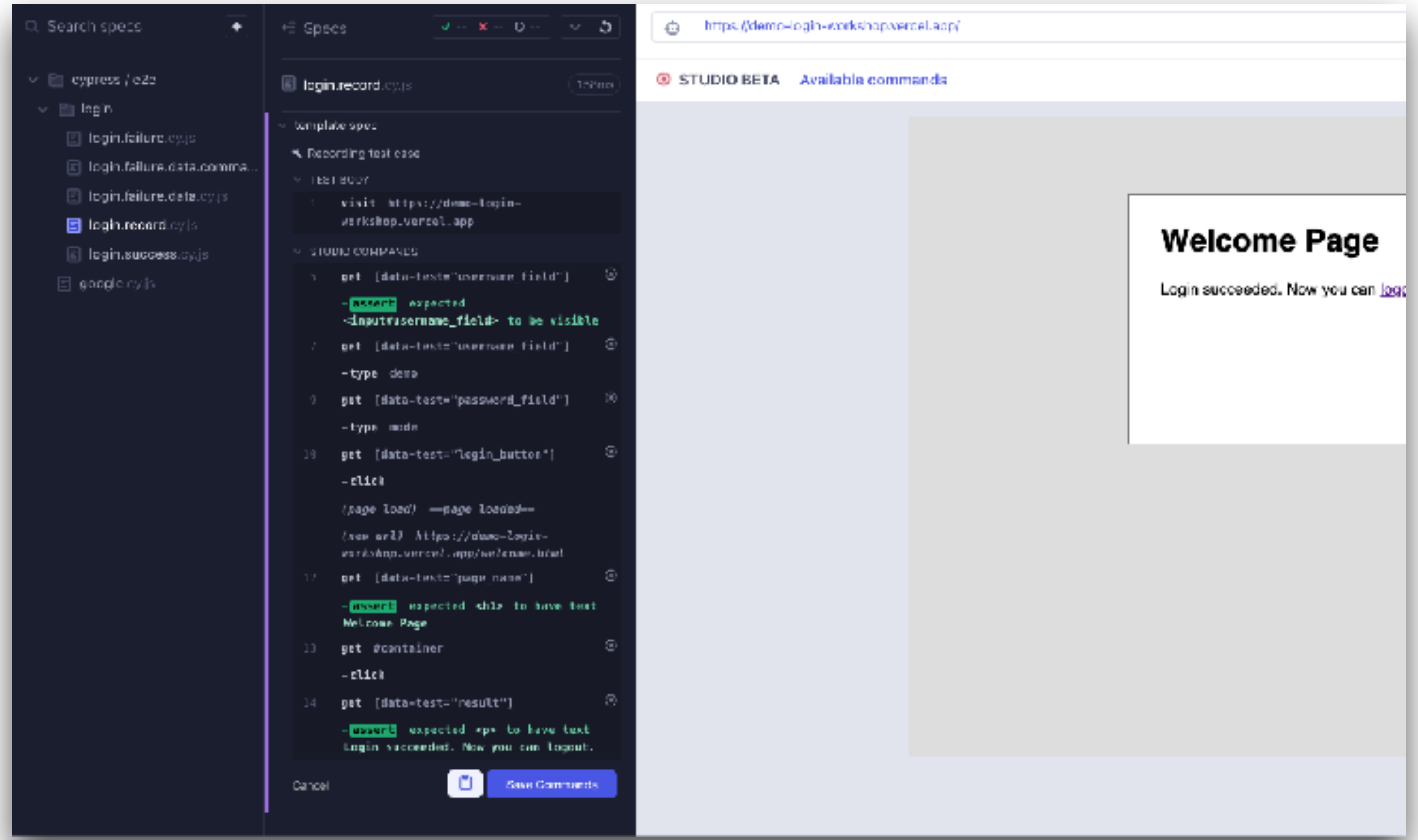
⚠ **Experimental**

Cypress Studio is an experimental feature and can be enabled by adding the experimentalStudio attribute to your Cypress configuration.

```
{
  e2e: {
    experimentalStudio: true
  }
}
```

https://docs.cypress.io/guides/references/cypress-studio

# Restart Cypress UI



```
template spec
  Recording test case
    TEST BODY
      visit  https://demo-login-workshop.vercel.app
    STUDIO COMMANDS
      get  [data-test="username_field"]
      -assert  expected <input#username_field> to be visible
      get  [data-test="username_field"]
      -type  demo
      get  [data-test="password_field"]
      -type  mode
      get  [data-test="login_button"]
      -click
      (page load)  --page loaded--
      (new url)  https://demo-login-workshop.vercel.app/welcome.html
      get  [data-test="page_name"]
      -assert  expected <h1> to have text Welcome Page
      get  #container
      -click
      get  [data-test="result"]
      -assert  expected <p> to have text Login succeeded. Now you can logout.
```

Welcome Page

Login succeeded. Now you can logo

## Create empty test case

```
describe("template spec", () => {
  it("Recording test case", { tags: ["login"] }, () => {

   cy.visit("https://demo-login-workshop.vercel.app");

  });
});
```

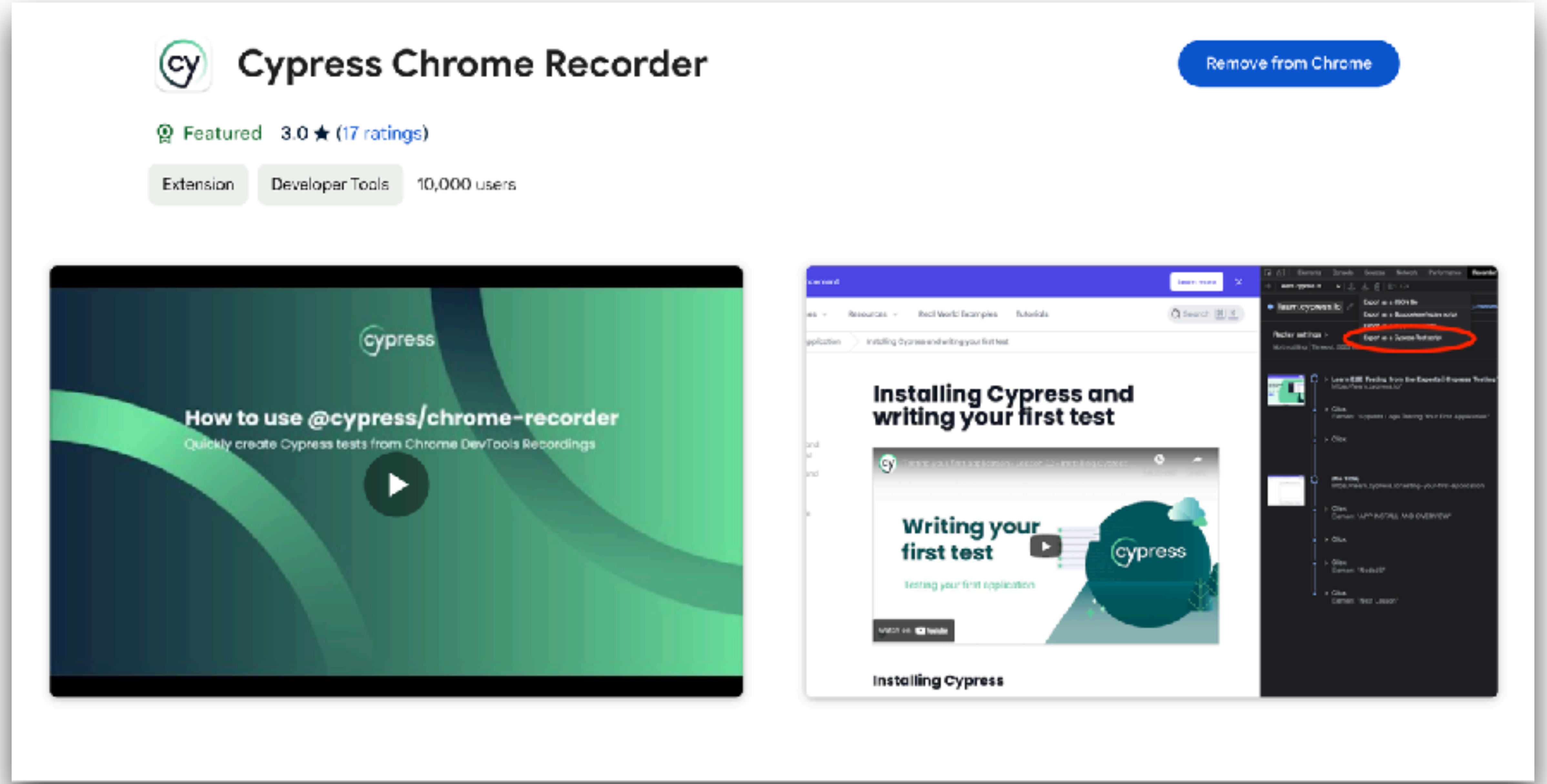
# Save commands

```
describe("template spec", () => {
  it("Recording test case", { tags: ["login"] }, () => {
    // Arrange
    cy.visit("https://demo-login-workshop.vercel.app");


    /* ==== Generated with Cypress Studio ==== */
    cy.get('[data-test="username_field"]').should('be.visible');
    cy.get('[data-test="username_field"]').type('demo');
    cy.get('[data-test="password_field"]').type('mode');
    cy.get('[data-test="login_button"]').click();
    cy.get('[data-test="page_name"]').should('have.text', 'Welcome Page');
    cy.get('#container').click();
    cy.get('[data-test="result"]').should('have.text', 'Login succeeded. Now you can logout.');
    /* ==== End Cypress Studio ==== */
  });
});
```

## Working with Chrome Recorder

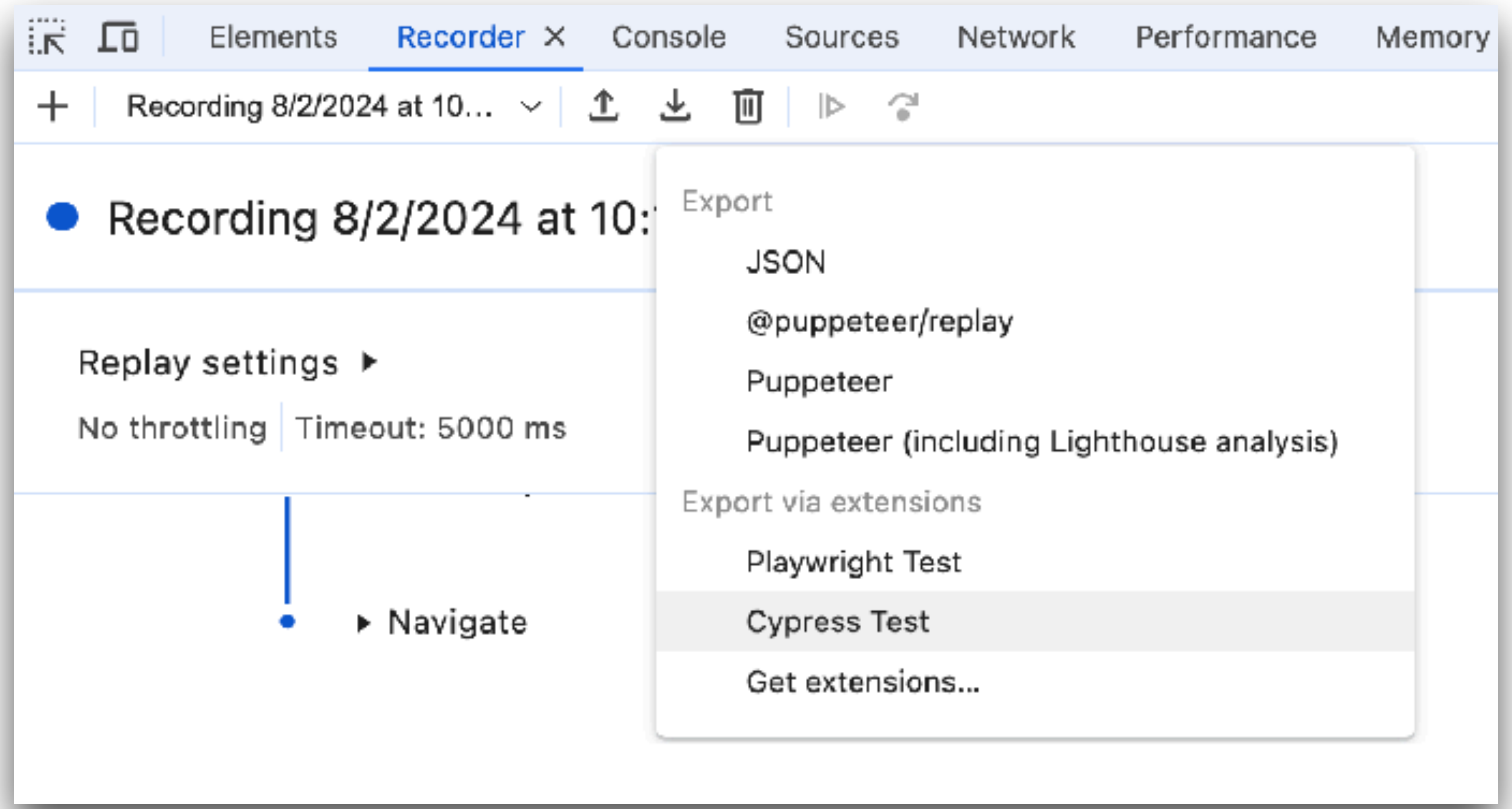
https://developer.chrome.com/docs/devtools/recorder

# Cypress Chrome Recorder


Cypress Chrome Recorder
Remove from Chrome
Featured 3.0 ★ (17 ratings)
Extension
Developer Tools
10,000 users
cypress
How to use @cypress/chrome-recorder
Quickly create Cypress tests from Chrome DevTools Recordings
Installing Cypress and writing your first test
Writing your first test
cypress
Installing Cypress

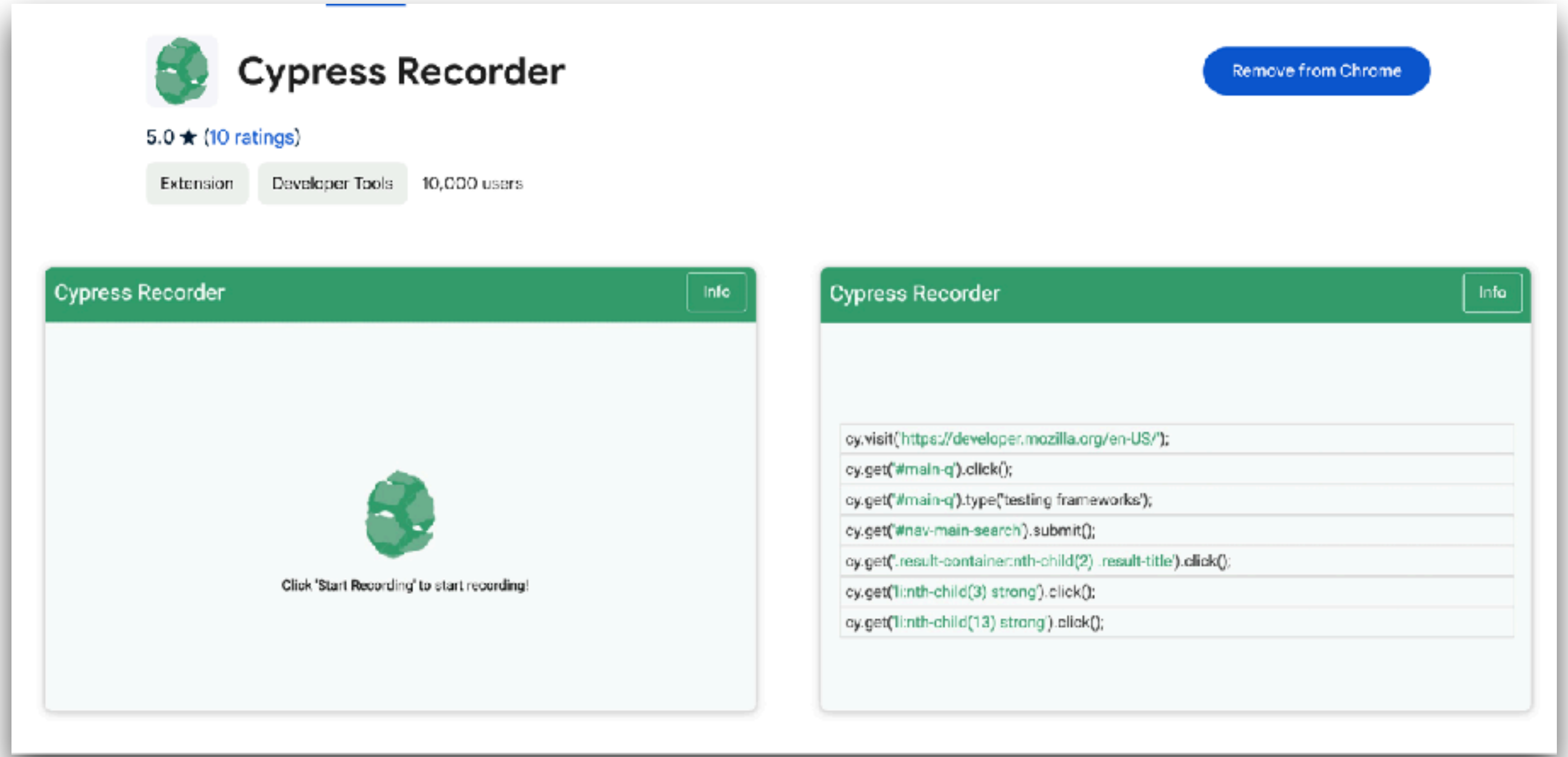
https://chromewebstore.google.com/detail/cypress-chrome-recorder/fellcphjglholofndfmmjmheedhomgin

# Cypress Chrome Recorder



https://chromewebstore.google.com/detail/cypress-chrome-recorder/fellcphjglholofndfmmjmheedhomgin

# Cypress Recorder



```
cy.visit('https://developer.mozilla.org/en-US/');
cy.get('#main-q').click();
cy.get('#main-q').type('testing frameworks');
cy.get('#nav-main-search').submit();
cy.get('.result-container:nth-child(2) .result-title').click();
cy.get('li:nth-child(3) strong').click();
cy.get('li:nth-child(13) strong').click();
```

https://chromewebstore.google.com/detail/cypress-recorder/glcapdcacdfkokcmicllhcjigeodacab

## Workshop

## Try by yourself

https://demo-login-workshop.vercel.app

1
Design test case

2
Write your test case

## Start with design test cases

# Test Cases

| Case name | Input | | Expected Result |
|---|---|---|---|
| | **Username** | **Password** | |
| Login success | demo | mode | Show welcome page |
| Login fail case A | demo | mode2 | Show error page |
| Login fail case B | demo2 | mode | Show error page |
| Login fail case C | demo2 | mode2 | Show error page |

## Write test cases

Success cases

Failure cases

## Login success

```
describe('template spec', () => {
  it('passes', () => {
    // Arrange
    cy.visit('https://demo-login-workshop.vercel.app')

    // Act
    cy.get('#username_field').should('be.visible').type('demo')
    cy.get('#password_field').should('be.visible').type('mode')
    cy.get('#login_button').should('be.visible').click()

    // Assert
    cy.contains('Login succeeded. Now you can logout.')
      .should('be.visible')

    cy.get('#container > h1').contains('Welcome Page').should('be.visible')
    cy.get('#container > p').contains('Login succeeded. Now you can 
logout.').should('be.visible')
  })
})
```

## Login failure

```
describe('Login fail spec', () => {
  it('Fail with wrong username', () => {
    // Arrange
    cy.visit('https://demo-login-workshop.vercel.app')

    // Act
    cy.get('#username_field').should('be.visible').type('demo2')
    cy.get('#password_field').should('be.visible').type('mode')
    cy.get('#login_button').should('be.visible').click()

    // Assert
    cy.get('#container > h1').contains('Error Page').should('be.visible')
    cy.get('#container > p').contains('Login failed. Invalid user name and/or
password.').should('be.visible')
  })
```

## Data Driven Testing

Use data from **fixtures**
Read data from test cases

Test case

Login-failures.json

# File login-failures.json

List of test datas

```
[
    {
        "test_case": "Fail with wrong username",
        "usename": "demo2",
        "password": "mode"
    },
    {
        "test_case": "Fail with wrong password",
        "usename": "demo",
        "password": "mode2"
    }
]
```

# Login failure

## Read data from fixtures

```
import datas from "../fixtures/login-fails.json"
describe('Login fail spec', () => {

  datas.forEach( data => {

    it(data.test_case, () => {
      // Arrange
      cy.visit('https://demo-login-workshop.vercel.app')

      // Act
      cy.get('#username_field').should('be.visible').type(data.usename)
      cy.get(‘#password_field').should('be.visible').type(data.password)
      cy.get('#login_button').should('be.visible').click()

    })
  })
})
```

## Remove duplication to command

# Duplication !!

```
import datas from "../fixtures/login-fails.json"

describe('Login fail spec', () => {

  datas.forEach( data => {

    it(data.test_case, () => {
      // Arrange
      cy.visit('https://demo-login-workshop.vercel.app')

      // Act
      cy.get('#username_field').should('be.visible').type(data.usename)
      cy.get(‘#password_field').should('be.visible').type(data.password)
      cy.get('#login_button').should('be.visible').click()

    })
  })
})
```

# Create a new command

File /support/commands.js

```
Cypress.Commands.add('login', (username, password) => {
    cy.get('#username_field').should('be.visible').type(username)
    cy.get('#password_field').should('be.visible').type(password)
    cy.get('#login_button').should('be.visible').click()
})
```

Use command from test case

```
it(data.test_case, () => {
      // Act
      cy.login(data.usename, data.password)
})
```

# Add attribute for <u>test</u> with cypress ?

data-cy, **data-test**, data-testid

https://docs.cypress.io/guides/references/best-practices#Selecting-Elements

# Add data-test attribute

File index.html

```
<table>
  <tr>
    <td><label for="username_field">User Name:</label></td>

    <td><input id="username_field" data-test="username_field"
               size="30" type="text">
    </td>
  </tr>
  <tr>
    <td><label for=“password_field">Password:</label></td>

    <td><input id="password_field" data-test="password_field"
               size="30" type=“password">
    </td>
  </tr>

</table>
```

# Create command with test-data

File /support/commands.js

```
Cypress.Commands.add('getBySel', (selector, ...args) => {

    return cy.get(`[data-test=${selector}]`, …args)

})
```

```
Cypress.Commands.add('login', (username, password) => {
    cy.getBySel('username_field').should('be.visible').type(username)
    cy.getBySel('password_field').should('be.visible').type(password)
    cy.getBySel('login_button').should('be.visible').click()
})
```

## Manage secret/sensitive data ?

## Secret/Sensitive Data

Credential

User and password

# Environment Variables ?

Different across developer machines
different across multiple environments
Change frequently and are highly dynamic

https://docs.cypress.io/guides/guides/environment-variables

# Example

Config **baseUrl** in cypress.config.js

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {

    baseUrl: 'https://www.google.com',

    setupNodeEvents(on, config) {
      // implement node event listeners here
    },
  },
});
```

## Use in test case

```
describe('Search with Google', () => {

    beforeEach(() => {
        // Arrange
        cy.intercept(
            {
              method: 'GET',
              pathname: '/complete/search*'
            },
            {
              body: []
            }
          ).as('search')

        cy.visit('/')

    })
```

# Add more env

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({

  env: {
    data: 'hello demo’,
  },

  e2e: {
    baseUrl: 'https://www.google.com',
    setupNodeEvents(on, config) {
      // implement node event listeners here
    },
  },
});
```

# Use in test case

```
/// <reference types="cypress" />

describe('Search with Google', () => {


    it('Search', () => {
        // Act
        cy.get(‘textarea[name="q"]')

        cy.get(‘[name="q"]').type(Cypress.env('data'))

        cy.get('[name="q"]').type('{enter}')
        // Assert
        cy.get('#result-stats')
        cy.contains('#result-stats', 'ผลการค้นหาประมาณ')
    })

})
```

## Run in command line

```
$CYPRESS_baseUrl= … cypress run
$cypress run --config baseUrl=…
```

https://docs.cypress.io/guides/guides/environment-variables

## Improve Readability of Tests

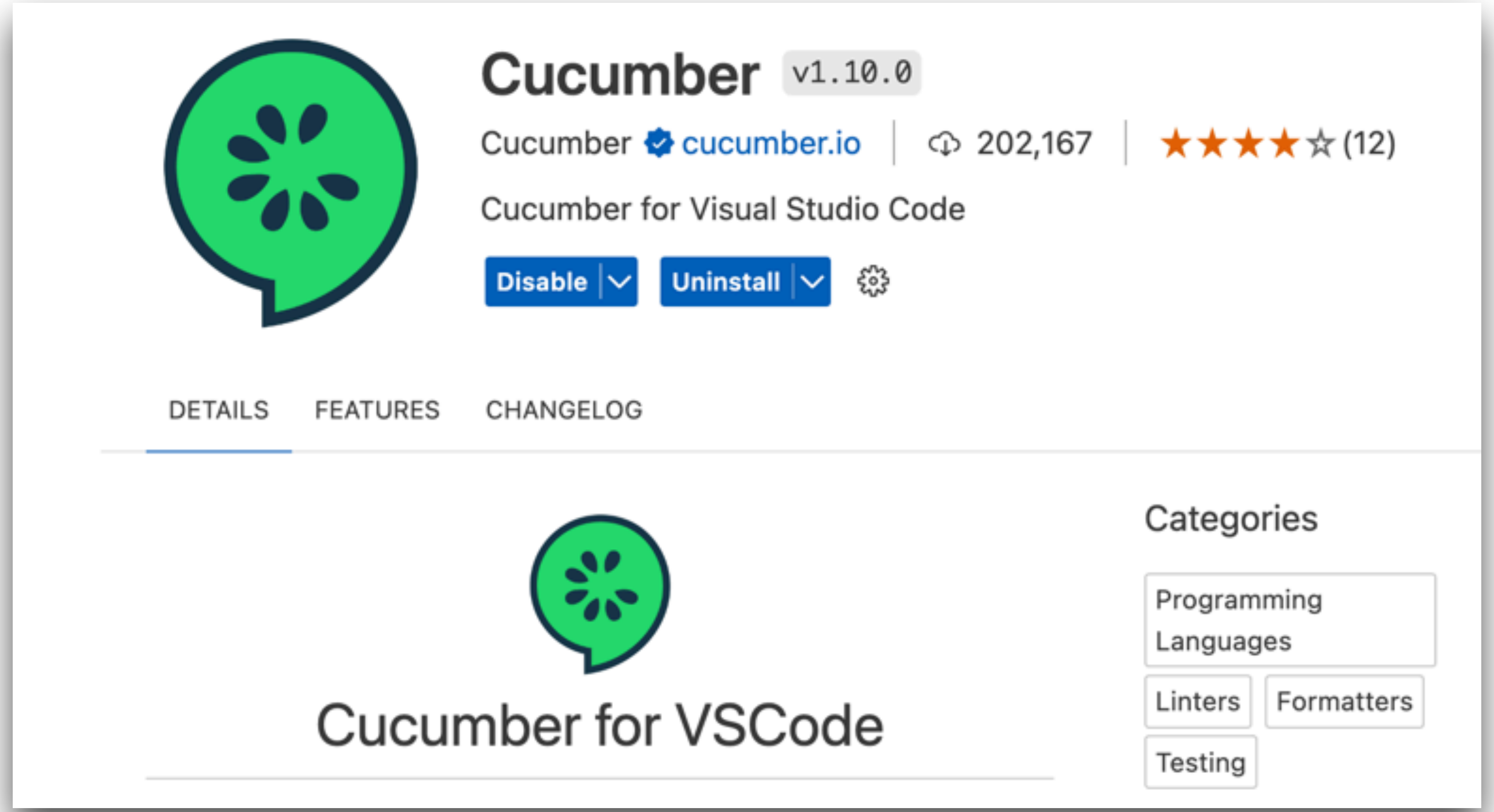
+ cucumber

# Cucumber

Feature files

Step definition files

# Cucumber plugin in VSCode



Cucumber v1.10.0

Cucumber cucumber.io | 202,167 | ★★★★☆ (12)

Cucumber for Visual Studio Code

Cucumber for VSCode

Categories

## Install

```
$npm i -D @badeball/cypress-cucumber-preprocessor
$npm i -D @cypress/browserify-preprocessor
```

https://github.com/badeball/cypress-cucumber-preprocessor

# Cypress.config.js

```
const { defineConfig } = require("cypress");
const {
  addCucumberPreprocessorPlugin,
} = require("@badeball/cypress-cucumber-preprocessor");
const {
  preprocessor,
} = require("@badeball/cypress-cucumber-preprocessor/browserify");

async function setupNodeEvents(on, config) {
  await addCucumberPreprocessorPlugin(on, config);
  on("file:preprocessor", preprocessor(config));
  return config;
}

module.exports = defineConfig({

  e2e: {
    specPattern: "**/*.feature",
    setupNodeEvents
  },
});
```

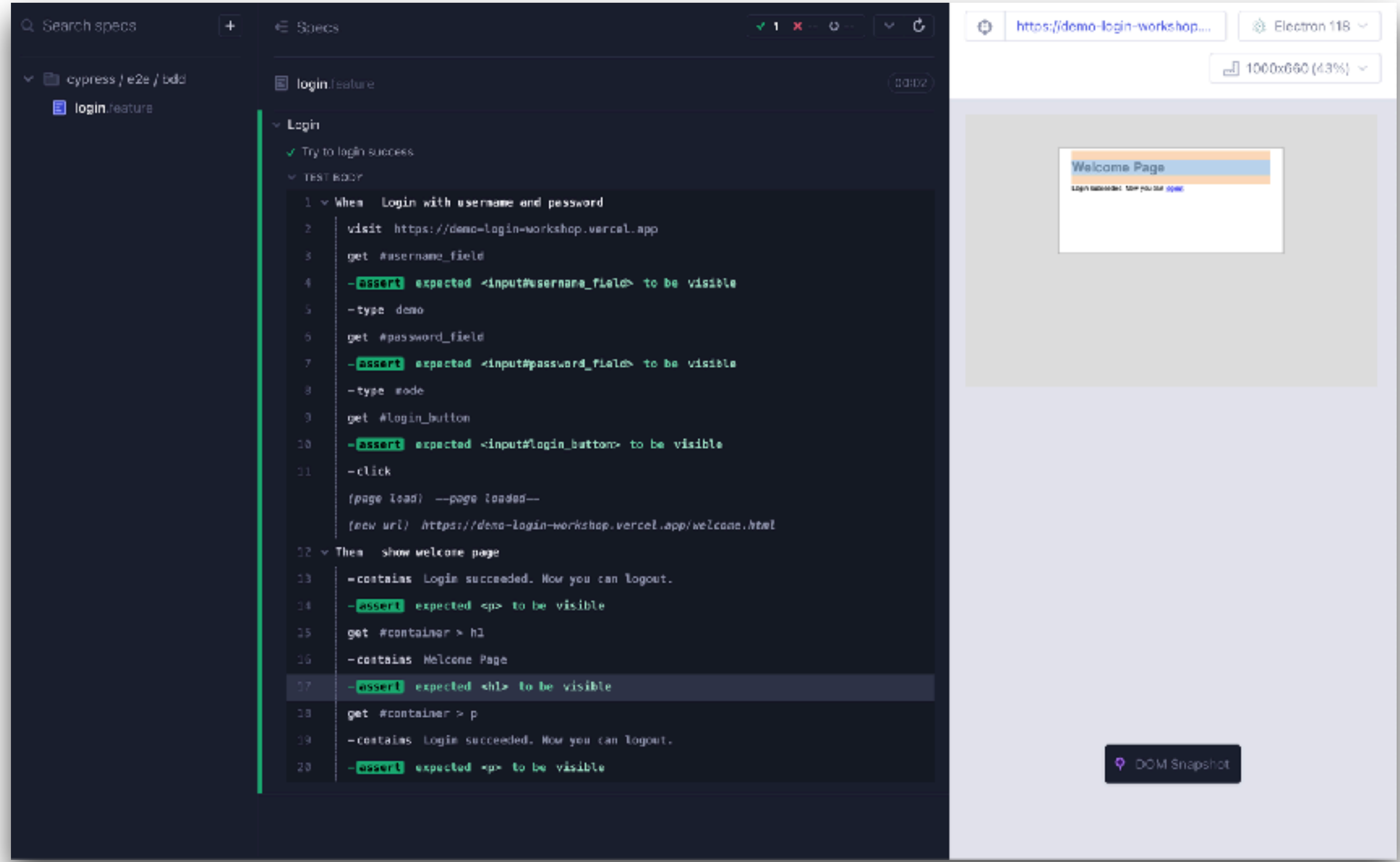
## login.feature

```
Feature: Login
    Scenario: Try to login success
        When Login with username and password
        Then show welcome page
```

# login.js (Steps definition)

```
import { When, Then } from "@badeball/cypress-cucumber-preprocessor";

When("Login with username and password", () => {
    cy.visit("https://demo-login-workshop.vercel.app");
    cy.get("#username_field").should("be.visible").type("demo");
    cy.get("#password_field").should("be.visible").type("mode");
    cy.get("#login_button").should("be.visible").click();
});

Then("show welcome page", () => {
    cy.contains("Login succeeded. Now you can logout.").should("be.visible");

    cy.get("#container > h1").contains("Welcome Page").should("be.visible");
    cy.get("#container > p")
      .contains("Login succeeded. Now you can logout.")
      .should("be.visible");
});
```

## Run test again !!



```
Login
✓ Try to login success
TEST BODY
When   Login with username and password
  visit  https://demo-login-workshop.vercel.app
  get  #username_field
  -assert  expected <input#username_field> to be visible
  -type  demo
  get  #password_field
  -assert  expected <input#password_field> to be visible
  -type  mode
  get  #login_button
  -assert  expected <input#login_button> to be visible
  -click
  (page load)  --page loaded--
  (new url)  https://demo-login-workshop.vercel.app/welcome.html
Then   show welcome page
  -contains  Login succeeded. Now you can logout.
  -assert  expected <p> to be visible
  get  #container > h1
  -contains  Welcome Page
  -assert  expected <h1> to be visible
  get  #container > p
  -contains  Login succeeded. Now you can logout.
  -assert  expected <p> to be visible
```

## Test Lifecycle

## Cypress Hooks

## Cypress Hooks

Life cycle for test cases
Manage state for test
Clean up state after test
Remove duplicate logics

# Cypress Hooks

| Name | Description |
|---|---|
| before | It runs once before starting the execution of first tests |
| after | It runs once after completion of all the tests |
| beforeEach | It runs before starting the execution of each of the tests |
| afterEach | It runs after finishing the execution of each of the tests |

https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests#Hooks

## Example

```
before(() => {
    // root-level hook
    // runs once before all tests
})

beforeEach(() => {
    // root-level hook
    // runs before every test block
})

afterEach(() => {
    // runs after each test block
})

after(() => {
    // runs once all tests are done
})
```

## Working with tags of tests

https://docs.cypress.io/guides/guides/command-line#cypress-run-tag-lt-tag-gt

# Cypress Grep

Filter tests using substring

```
$npm i -D @cypress/grep
```

https://www.npmjs.com/package/@cypress/grep

# Configuration (1)

File /supports/e2e.js

```
import './commands'

// Alternatively you can use CommonJS syntax:
// require('./commands')

const registerCypressGrep = require('@cypress/grep')
registerCypressGrep()
```

## Configuration (2)

File cypress.config.js

```
const { defineConfig } = require("cypress");

module.exports = defineConfig({
  e2e: {
    setupNodeEvents(on, config) {
      // implement node event listeners here
      require('@cypress/grep/src/plugin')(config);
    },
  },
});
```

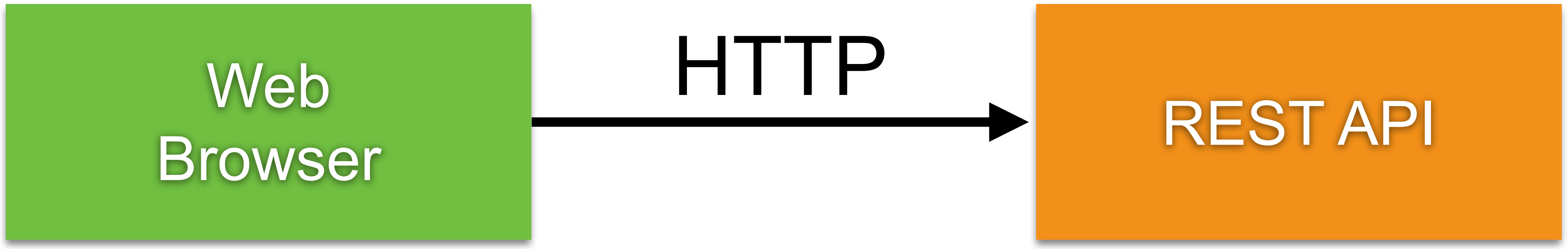
# Run with cypress grep

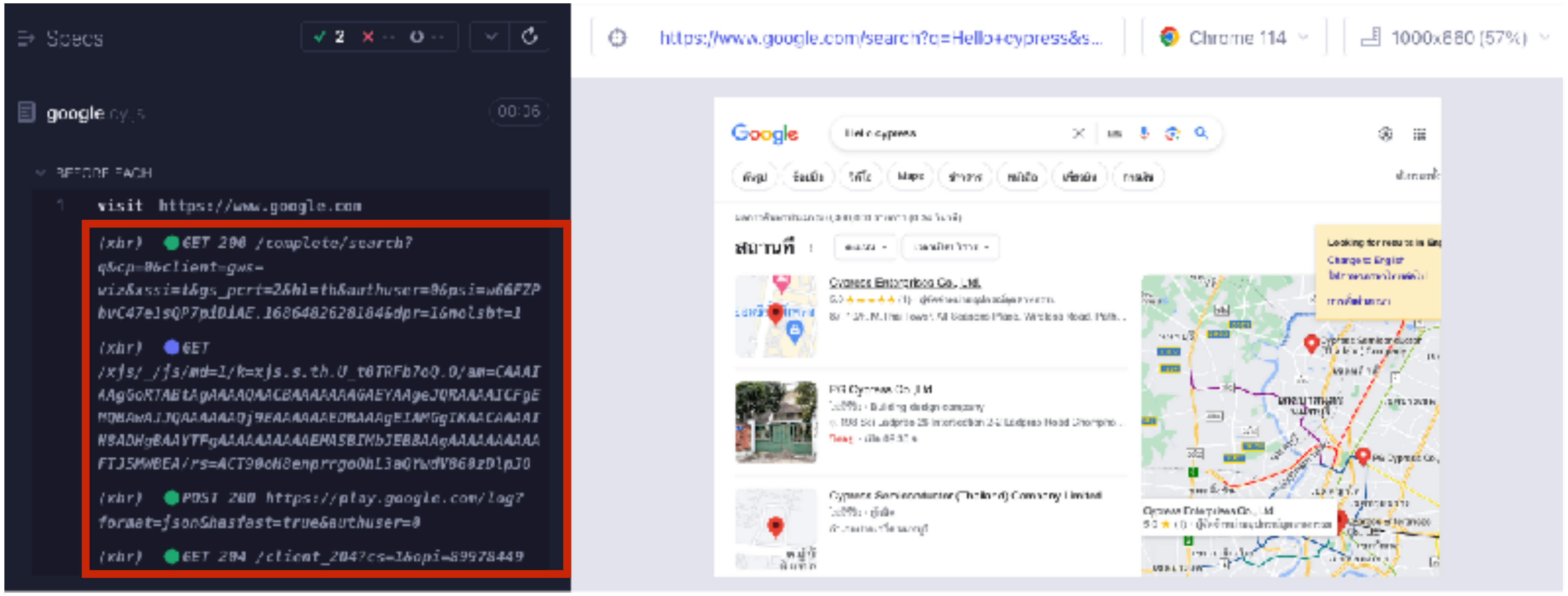
$cypress run --env grepTags=demo01

```
it("Success with load first page", {tags: ["demo01"]} , () => {
    // Assert
    cy.get("#APjFqb").should("be.visible");
    cy.get('[name="q"]').should("be.visible");
});
```

## Working with XHR (XML HTTP Request)

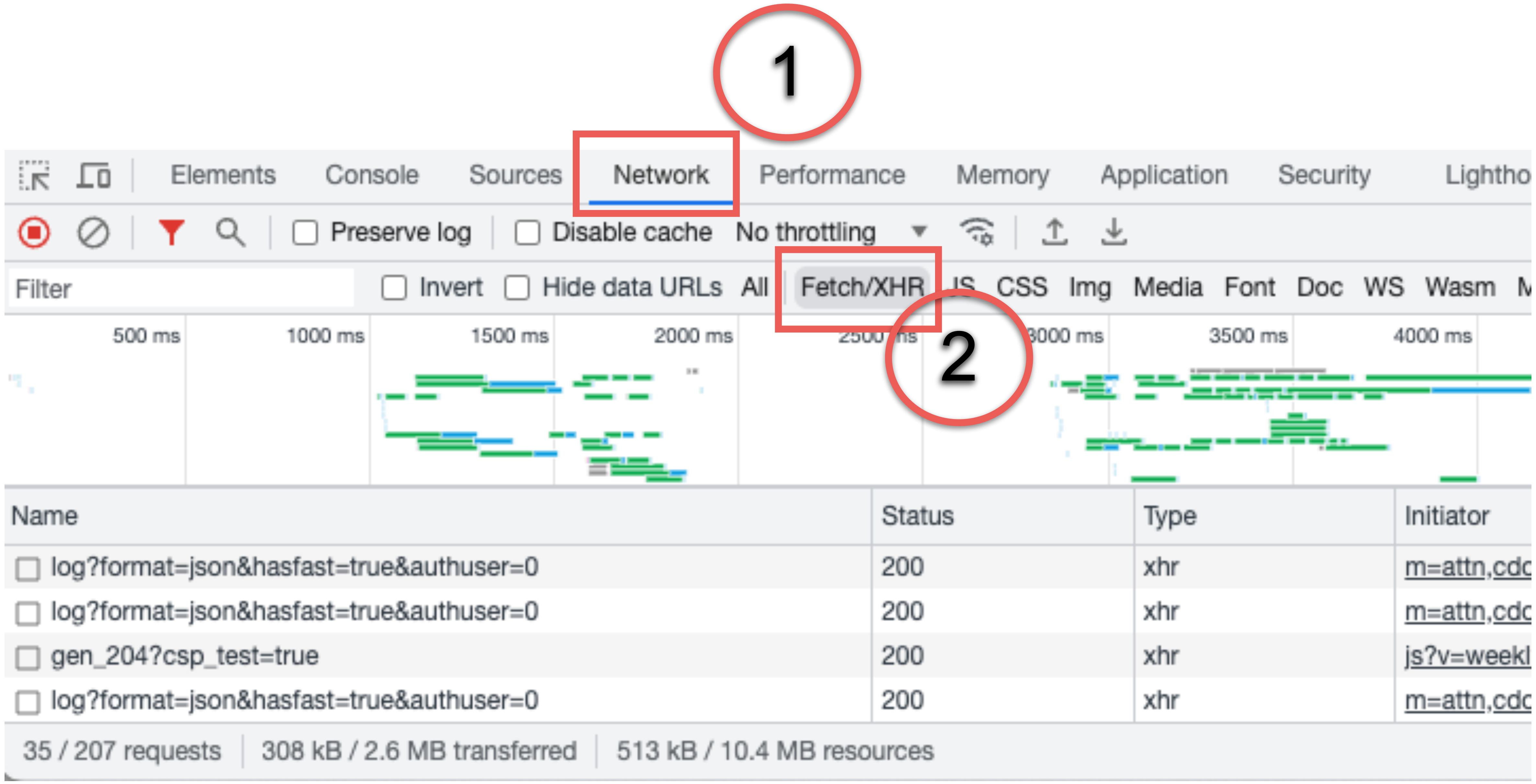
# XHR





```
1   visit  https://www.google.com
    (xhr)  GET 200 /complete/search?
    q&cp=0&client=gws-
    wiz&xssi=t&gs_pcrt=2&hl=th&authuser=0&psi=w66FZP
    bvC47elsQP7p1D1AE.1686482628184&dpr=1&nolsbt=1
    (xhr)  GET
    /xjs/_/js/md=1/k=xjs.s.th.U_t0TRFb7oQ.O/am=CAAAI
    AAg5oRTABtAgAAAAQAACBAAAAAAAGAEYAAgeJQRAAAAICFgE
    MQBAwAIJQAAAAAAQj9EAAAAAAEDBAAAgEIAMGgIKAACAAAAI
    N8ADHgBAAYTFgAAAAAAAAAAEMA5BIMbJEBBAAgAAAAAAAAAA
    FTJ5MWBEA/rs=ACT90oH8enprrgo0hL3aQYwdV868zD1pJ0
    (xhr)  POST 200 https://play.google.com/log?
    format=json&hasfast=true&authuser=0
    (xhr)  GET 204 /client_204?cs=1&opi=89978449
```

# XHR in Google Chrome

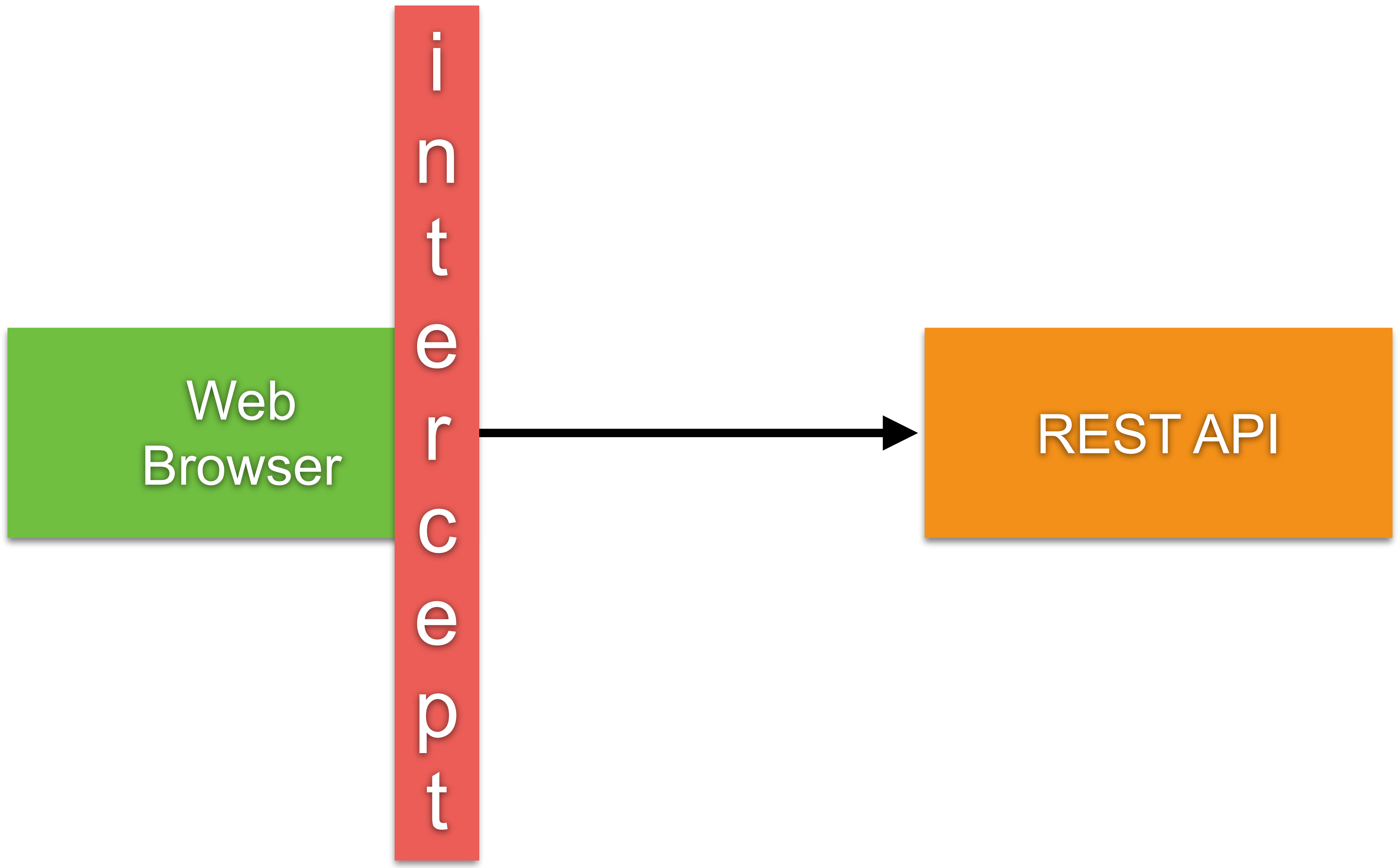


| Name | Status | Type | Initiator |
|---|---|---|---|
| log?format=json&hasfast=true&authuser=0 | 200 | xhr | m=attn,cdc |
| log?format=json&hasfast=true&authuser=0 | 200 | xhr | m=attn,cdc |
| gen_204?csp_test=true | 200 | xhr | js?v=weekl |
| log?format=json&hasfast=true&authuser=0 | 200 | xhr | m=attn,cdc |

35 / 207 requests | 308 kB / 2.6 MB transferred | 513 kB / 10.4 MB resources

# XHR with Cypress

Use network request + intercept in cypress
Stub out the backend from test



https://docs.cypress.io/guides/guides/network-requests

# Use cases for XHR with cypress

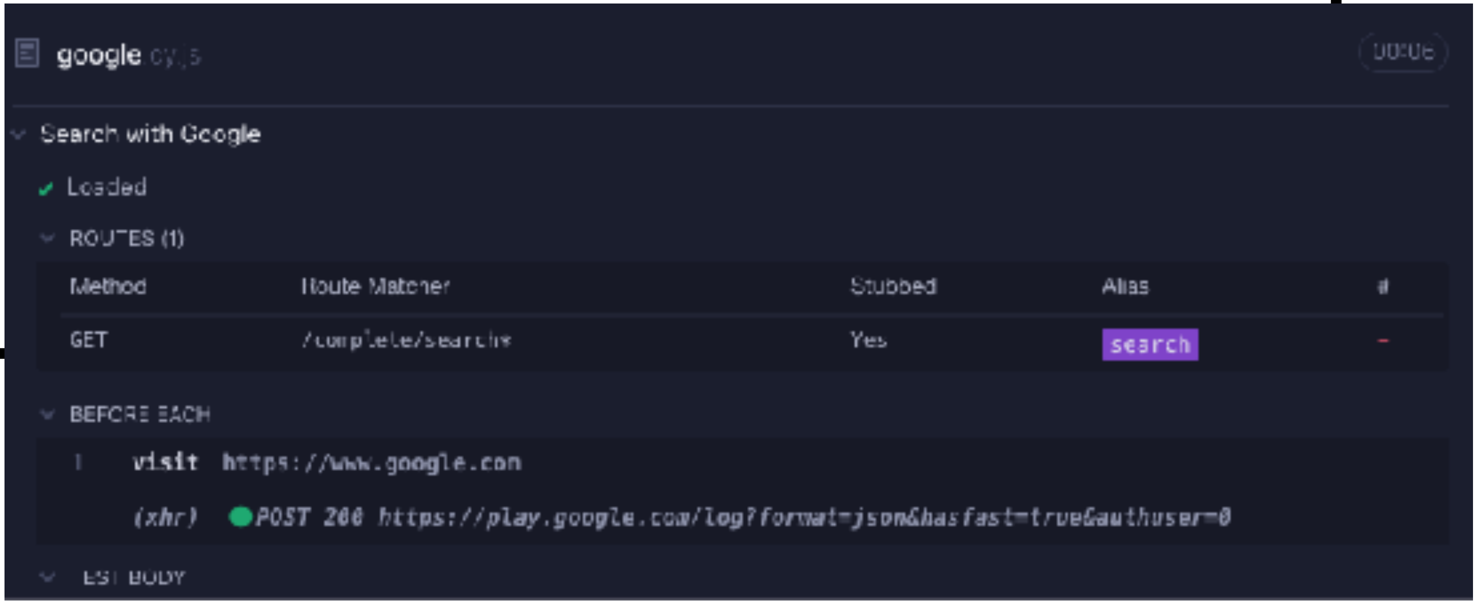
Assert request header/body
Stub response header/body
Delay response
Waiting for response

https://docs.cypress.io/guides/guides/network-requests

# XHR with Cypress

## Update file /e2e/google.cy.js

```
describe('Search with Google', () => {

    beforeEach(() => {
        // Arrange
        cy.intercept(
            {
              method: 'GET',
              pathname: '/complete/search*'
            },
            {
              body: []
            }
          ).as('search')

    })
```



google.cy.js 00:06

Search with Google

✓ Loaded

ROUTES (1)

| Method | Route Matcher | Stubbed | Alias | # |
|---|---|---|---|---|
| GET | /complete/search* | Yes | search | - |

BEFORE EACH

```
1   visit  https://www.google.com
    (xhr)  ●POST 200 https://play.google.com/log?format=json&hasfast=true&authuser=0
```

TEST BODY

## Routing

```
describe('Search with Google', () => {

    beforeEach(() => {

      cy.intercept(
           {
             method: 'GET',
             pathname: '/complete/search*'
           },
           {
             body: []
           }
         ).as('search')

    })
```

# Fixtures

Set of data located in file (/fixtures)

```
describe('Search with Google', () => {

    beforeEach(() => {

        cy.intercept(
             {
               method: 'GET',
               pathname: '/complete/search*'
             },
             {
               fixture: 'data.json',
             }
           ).as('search')

    })
```

# Waiting

## Wait for request and response

```
describe('Search with Google', () => {

    beforeEach(() => {
        cy.intercept(
            {
              method: 'GET',
              pathname: '/complete/search*'
            },
            {
              fixture: 'data.json',
            }
        ).as('search')
    })

    it('Loaded', () => {
        cy.wait(['@search'])
    })
```

# Delay response ?

## Update file /e2e/google.cy.js

```
describe('Search with Google', () => {

    beforeEach(() => {
        // Arrange
        cy.intercept(
            {
              method: 'GET',
              pathname: '/complete/search*'
            },
            {
              body: [],
              delay: 2000
            }
          ).as('search')
        cy.visit('https://www.google.com')
    })
```

# GenericStaticResponse

```
export interface GenericStaticResponse<Fixture, Body> {
  /**
   * Serve a fixture as the response body.
   */
  fixture?: Fixture
  /**
   * Serve a static string/JSON object as the response body.
   */
  body?: Body
  /**
   * HTTP headers to accompany the response.
   * @default {}
   */
  headers?: { [key: string]: string }
  /**
   * The HTTP status code to send.
   * @default 200
   */
  statusCode?: number
  /**
   * If 'forceNetworkError' is truthy, Cypress will destroy the browser connection
   * and send no response. Useful for simulating a server that is not reachable.
   * Must not be set in combination with other options.
   */
  forceNetworkError?: boolean
  /**
   * Kilobytes per second to send 'body'.
   */
  throttleKbps?: number
  /**
   * Milliseconds to delay before the response is sent.
   */
  delay?: number
}
```

## Working with Docker

https://docs.cypress.io/examples/docker

# Dockerfile

```dockerfile
FROM cypress/browsers
WORKDIR /opt/app
COPY . /opt/app
RUN --mount=type=cache,target=/root/.cache/npx npx cypress install
CMD ["npx", "cypress", "run"]
```

https://hub.docker.com/r/cypress/browsers/

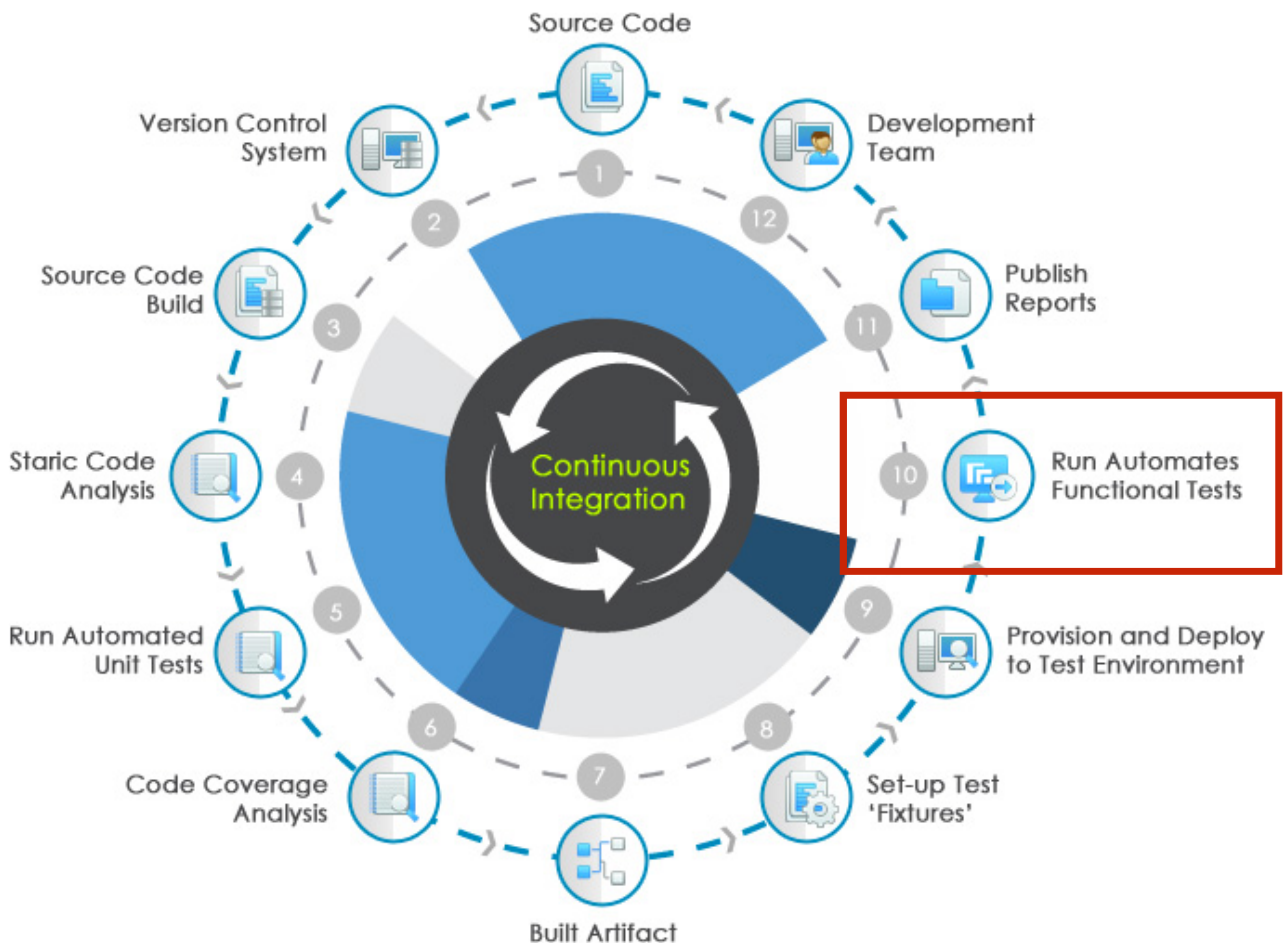
# Docker-compose.yml

$docker compose up testing --abort-on-container-exit --build

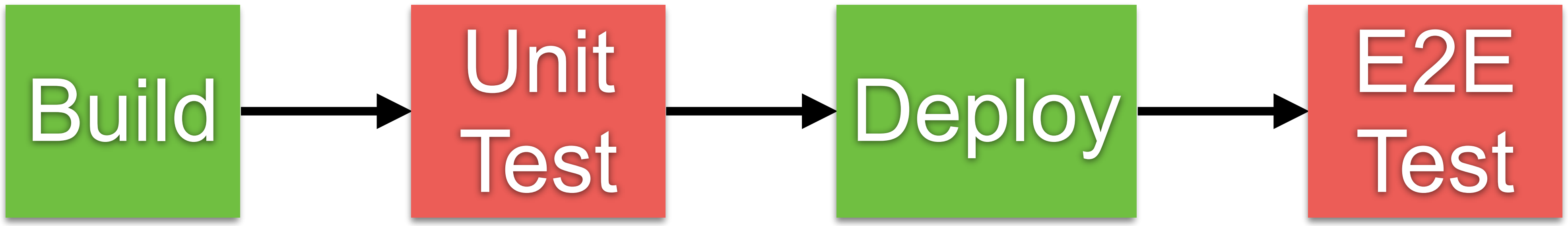
```yaml
services:
  testing:
    build:
      context: .
      dockerfile: Dockerfile
```

## Continuous Integration

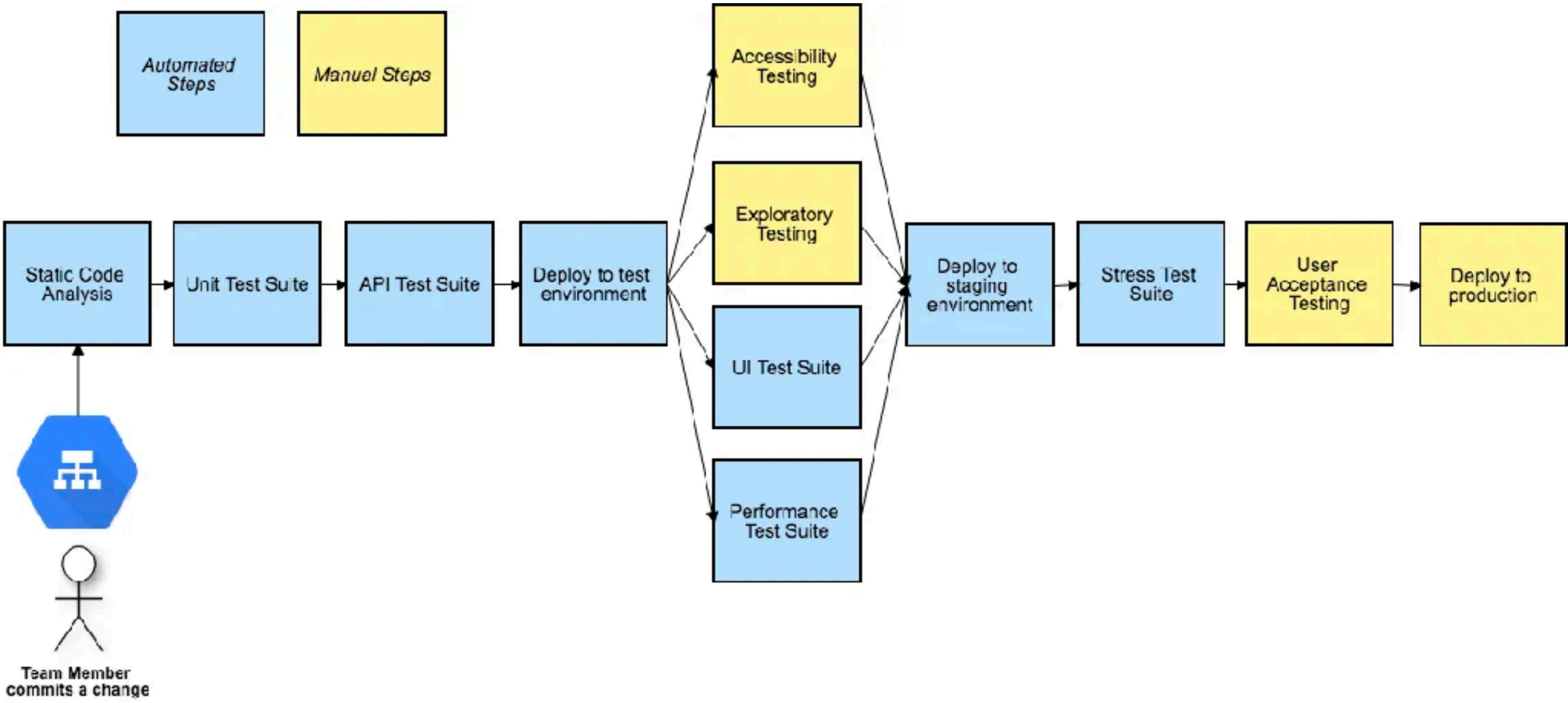
# Continuous integration



# Design your pipeline

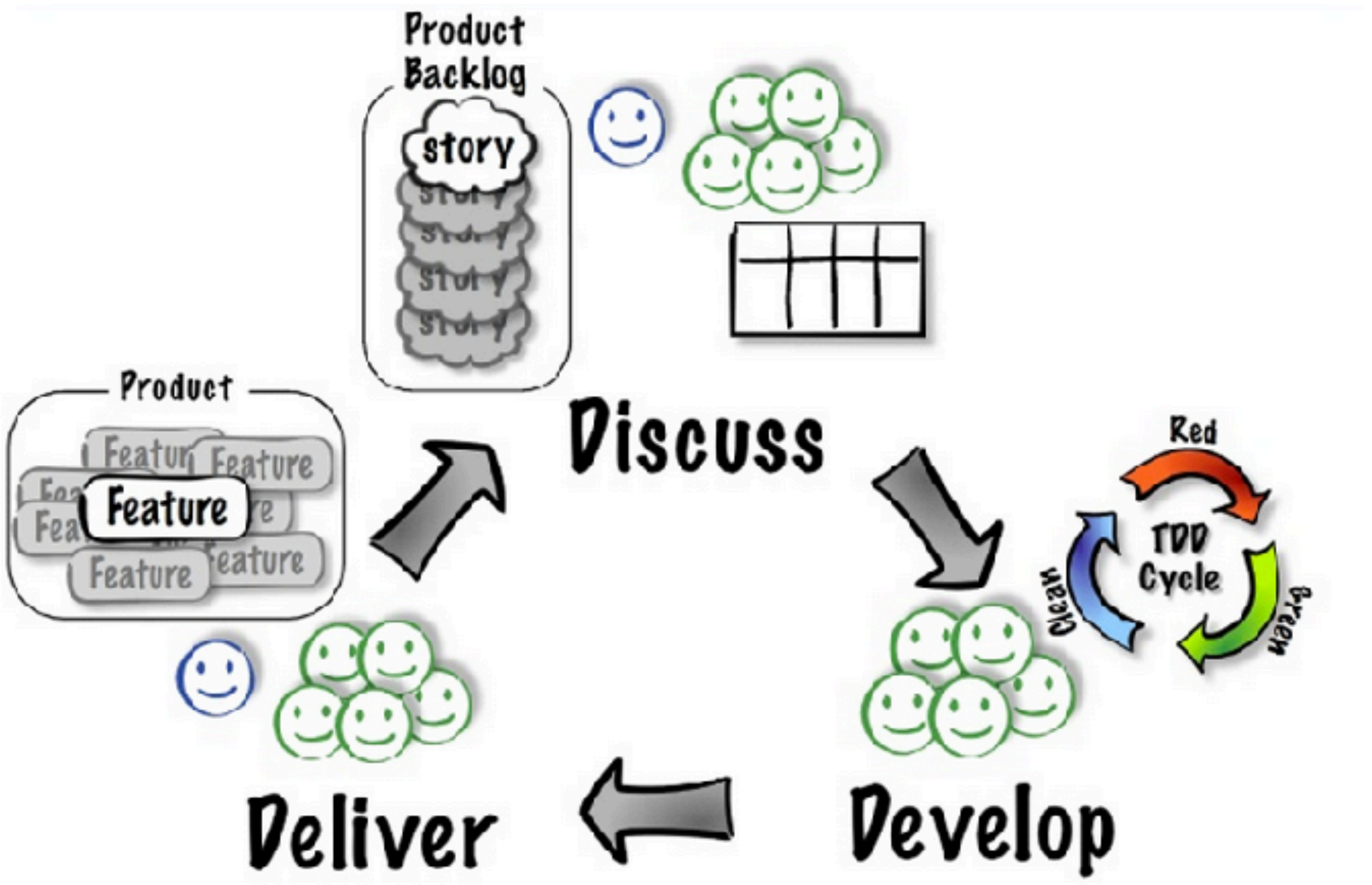


# Design your pipeline/process



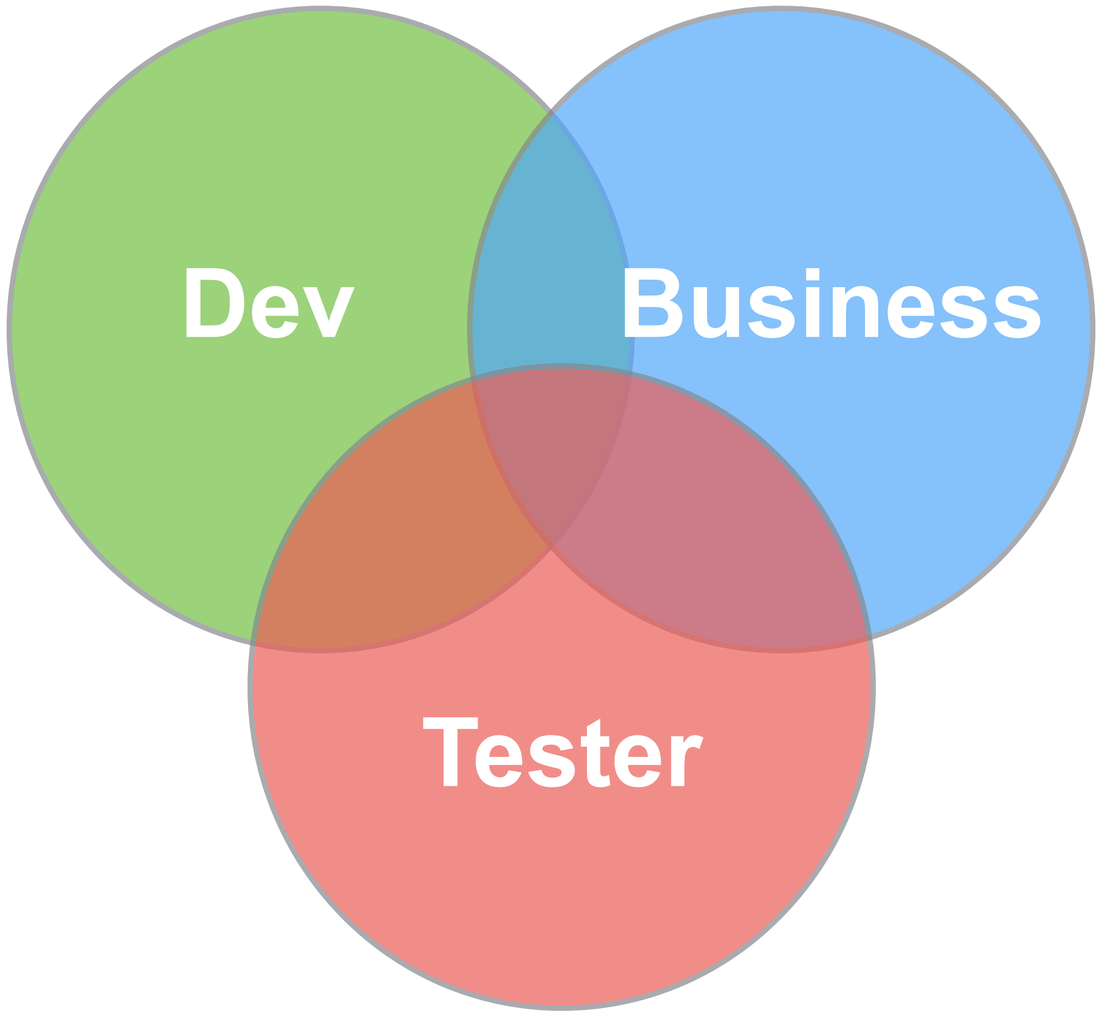
## THINK before coding

# Acceptance Test-Driven Development



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

# Acceptance Test-Driven Development



## Acceptance Tests

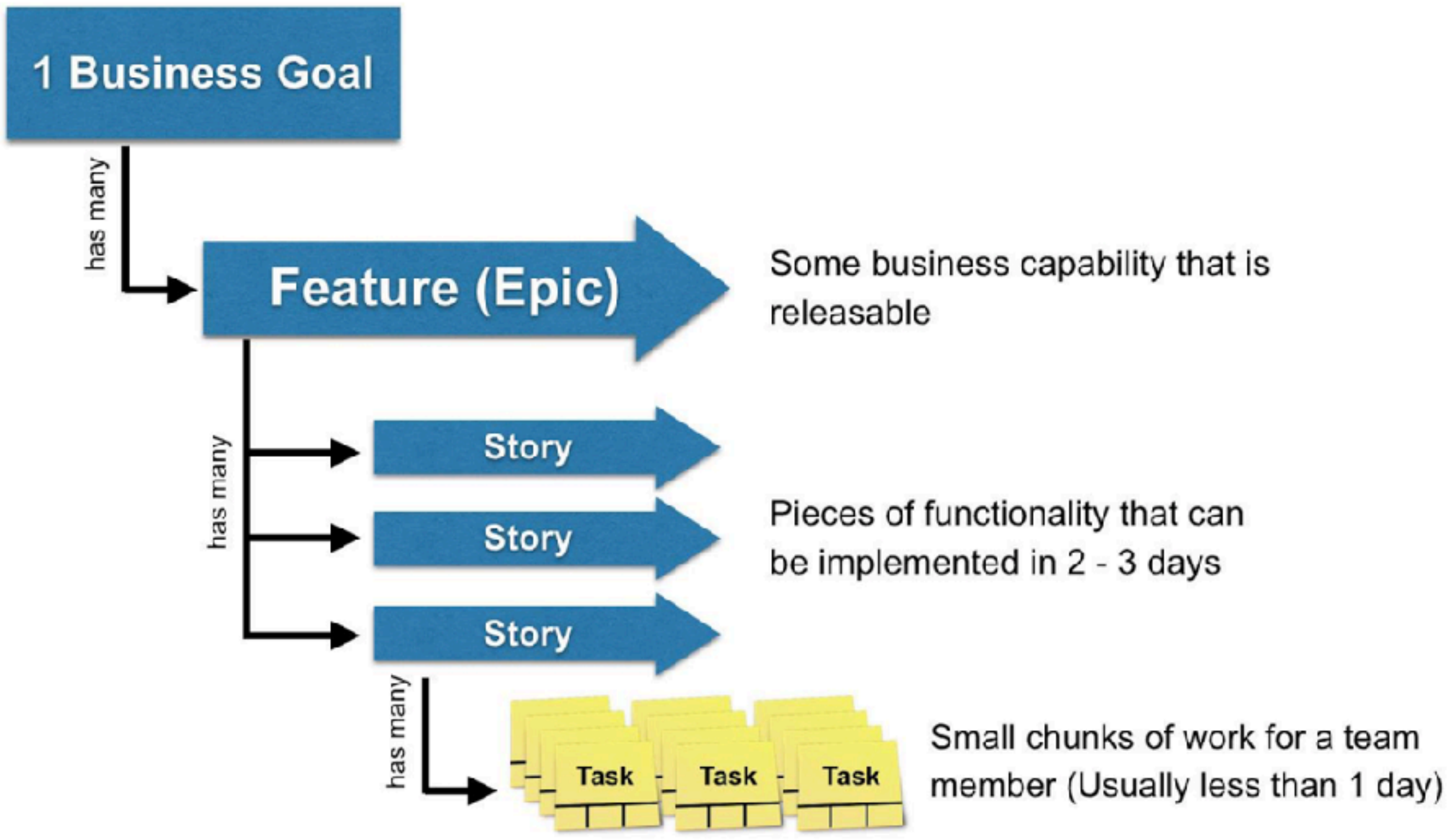
## =

## Business Criteria

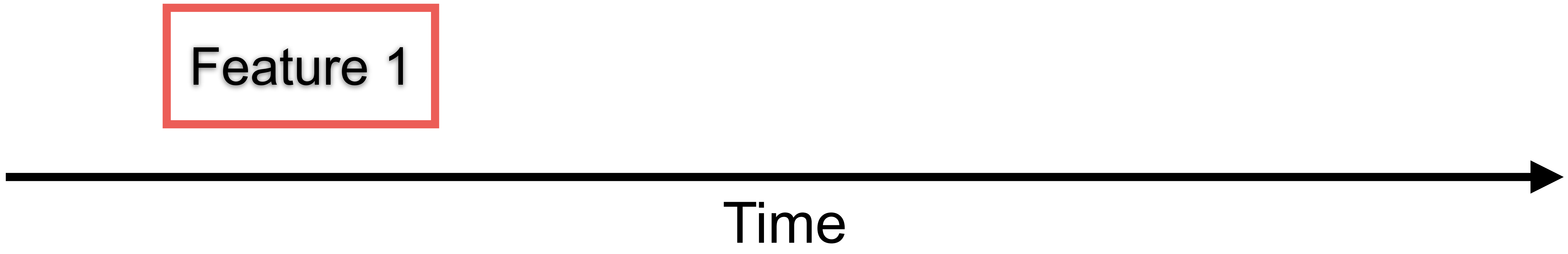
## +

## Examples (data)

# Work break down

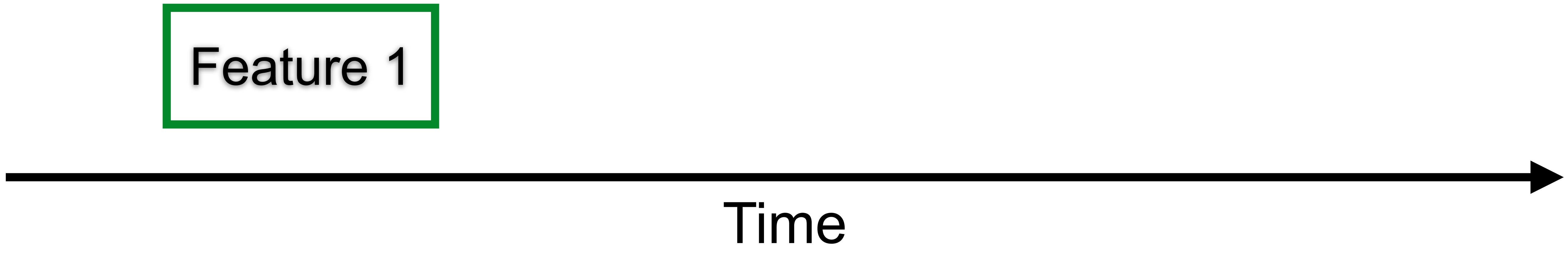

1 Business Goal
has many
Feature (Epic)
Some business capability that is releasable
has many
Story
Story
Story
Pieces of functionality that can be implemented in 2 - 3 days
has many
Task
Task
Task
Small chunks of work for a team member (Usually less than 1 day)

## Iterative and incremental process



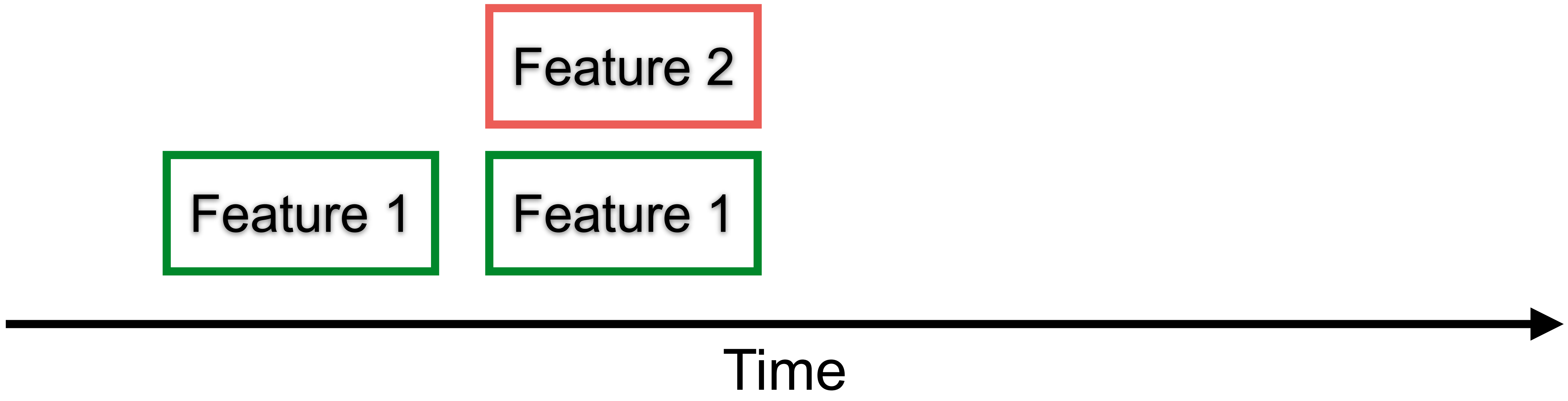
## Iterative and incremental process

Done = coded and tested



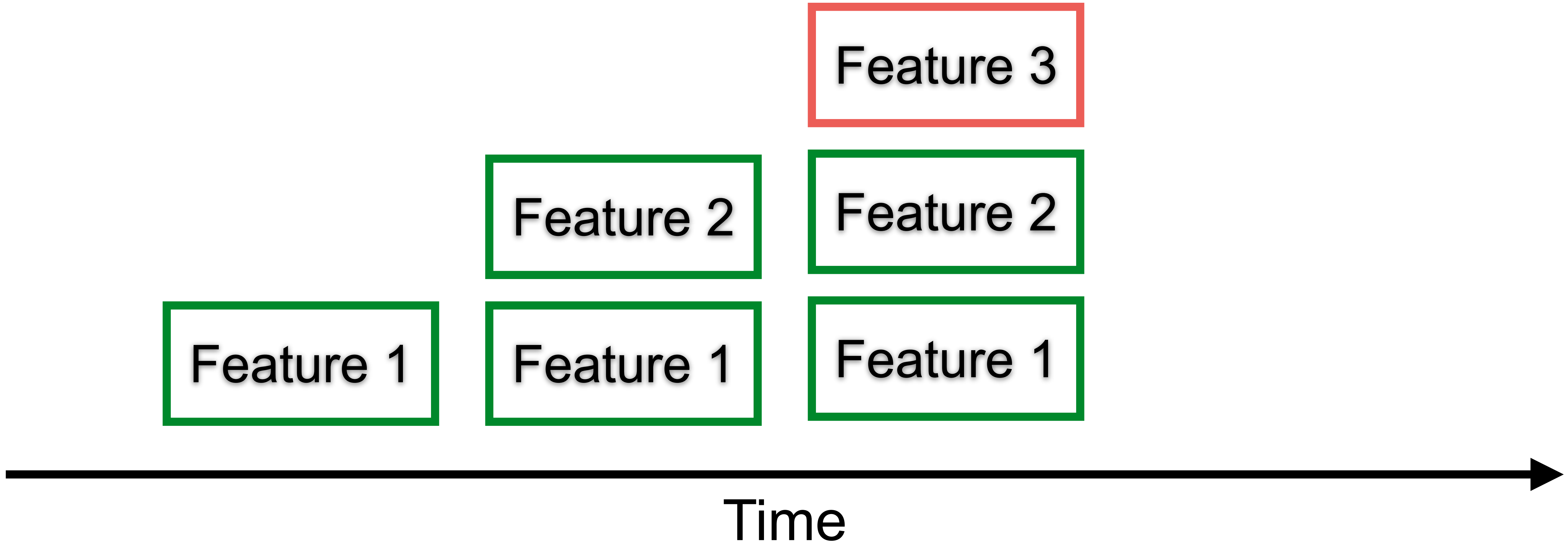
# Iterative and incremental process

Done = coded and tested



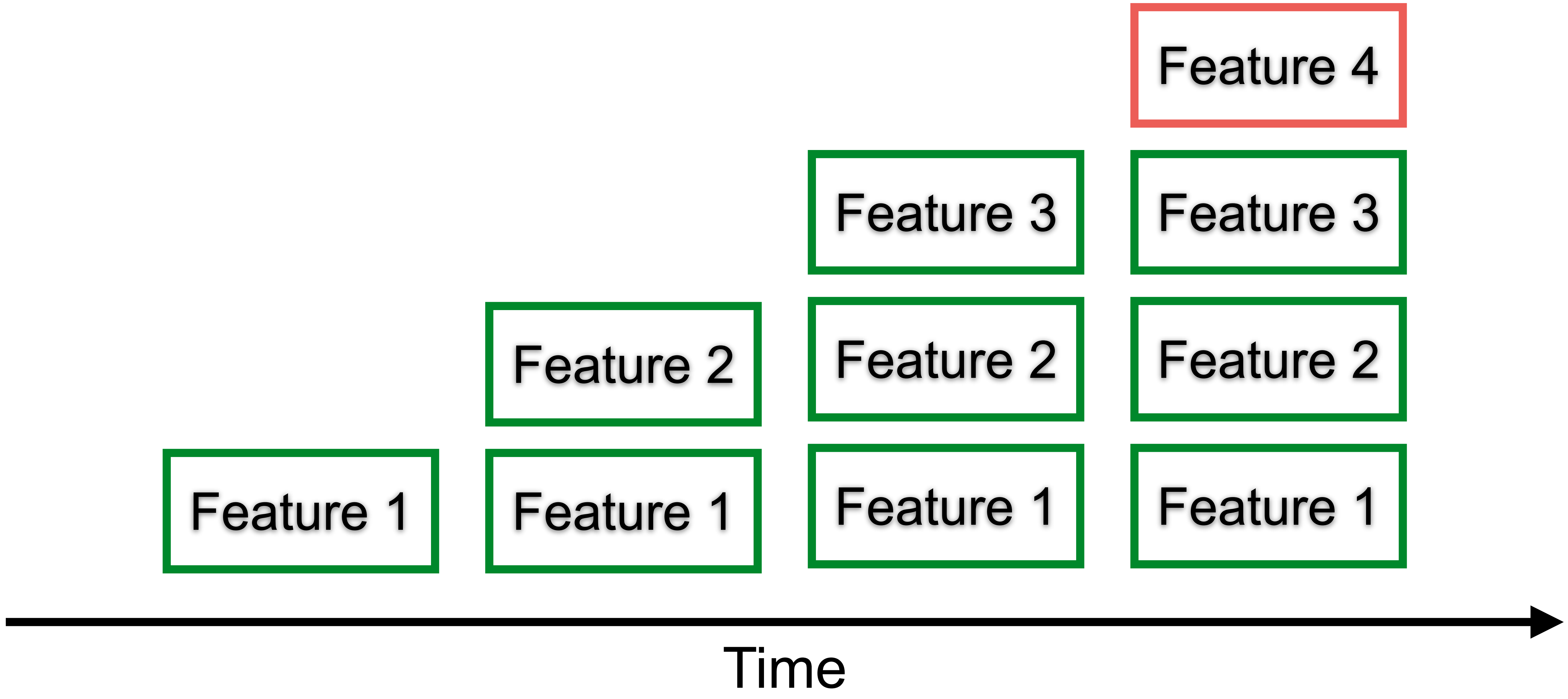
# Iterative and incremental process

Done = coded and tested



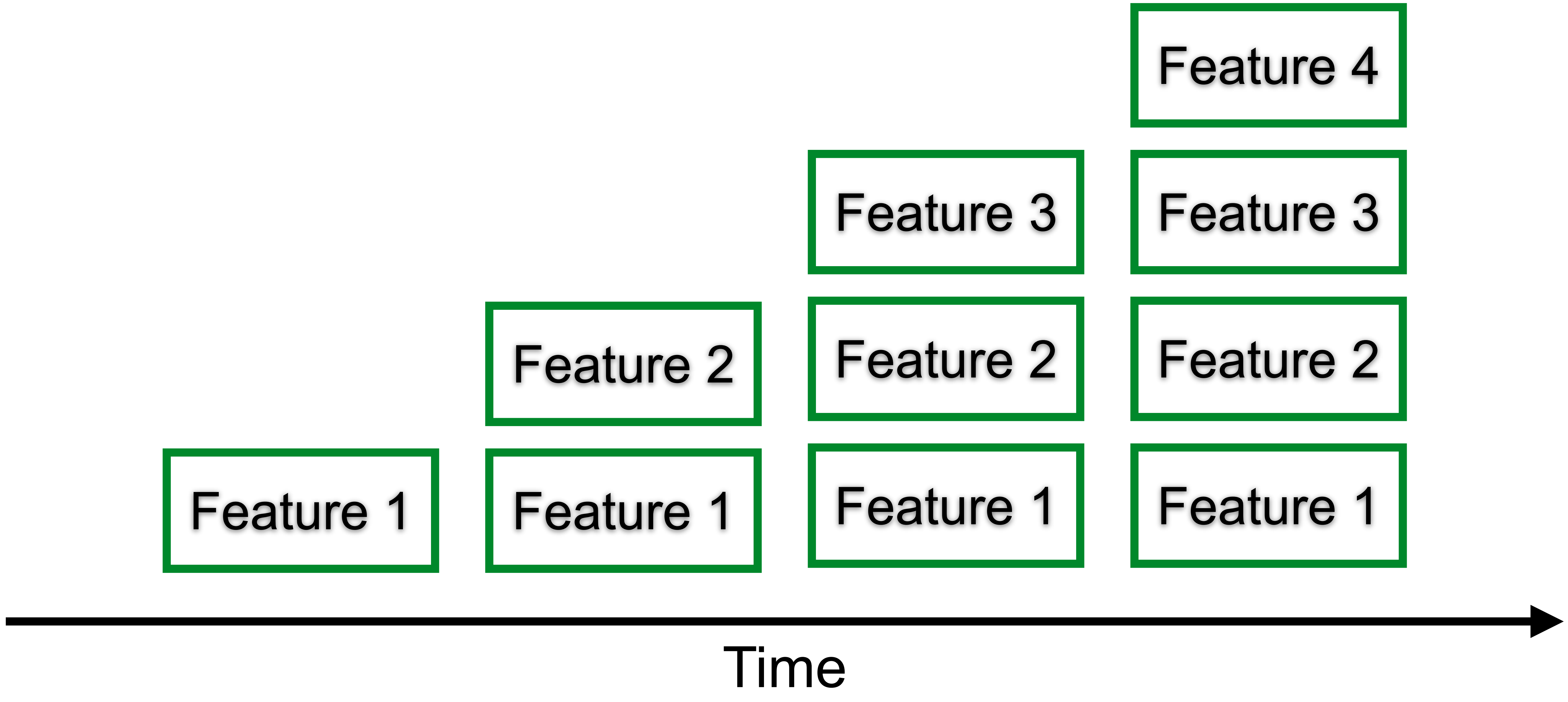
# Iterative and incremental process

Done = coded and tested



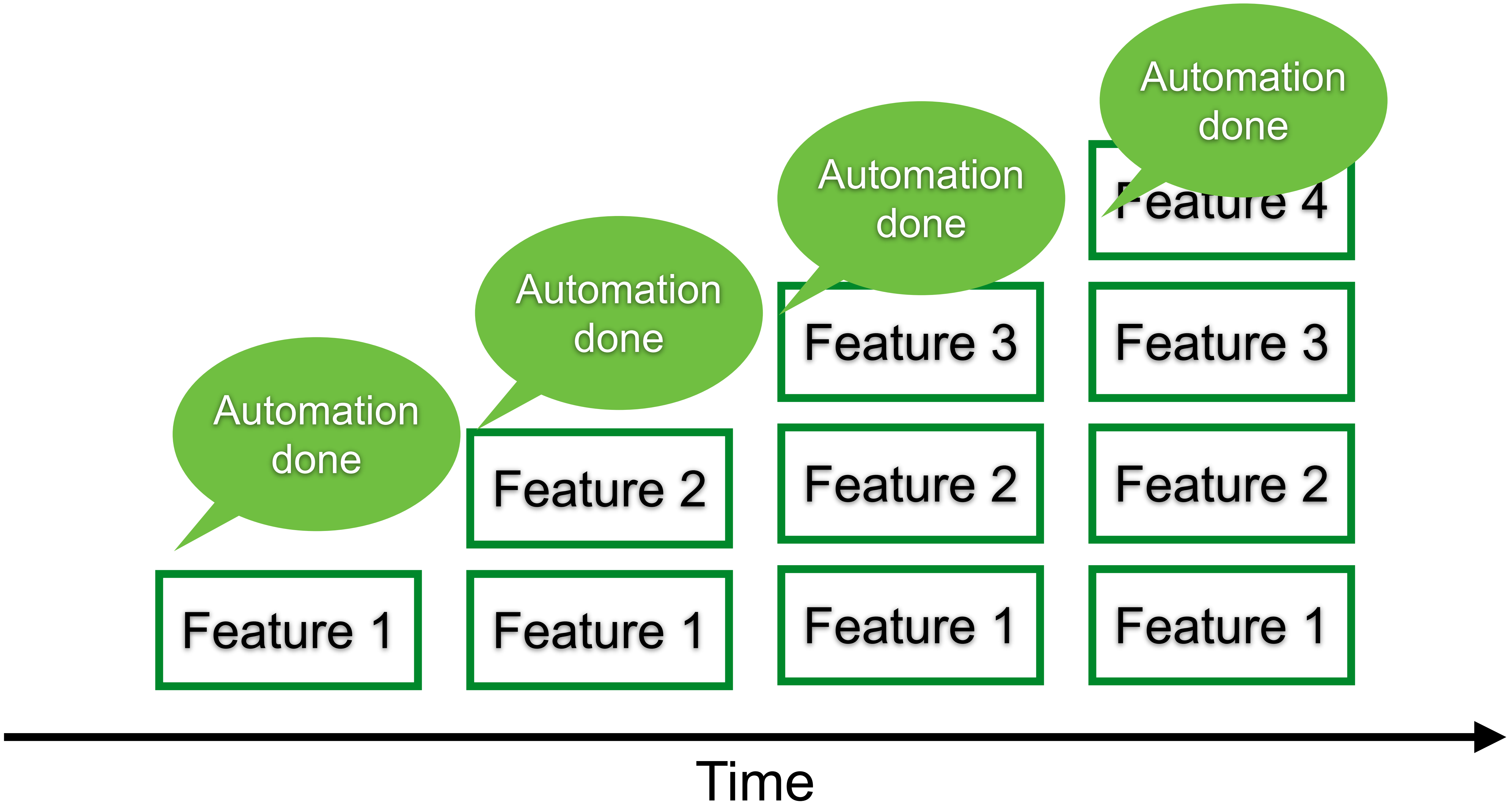
# Iterative and incremental process

Done = coded and tested



# Iterative and incremental process

Done = coded and tested


Automation done
Automation done
Automation done
Automation done
Feature 1
Feature 2
Feature 1
Feature 3
Feature 2
Feature 1
Feature 4
Feature 3
Feature 2
Feature 1
Time