

Automated Testing with Cypress



Automated Testing



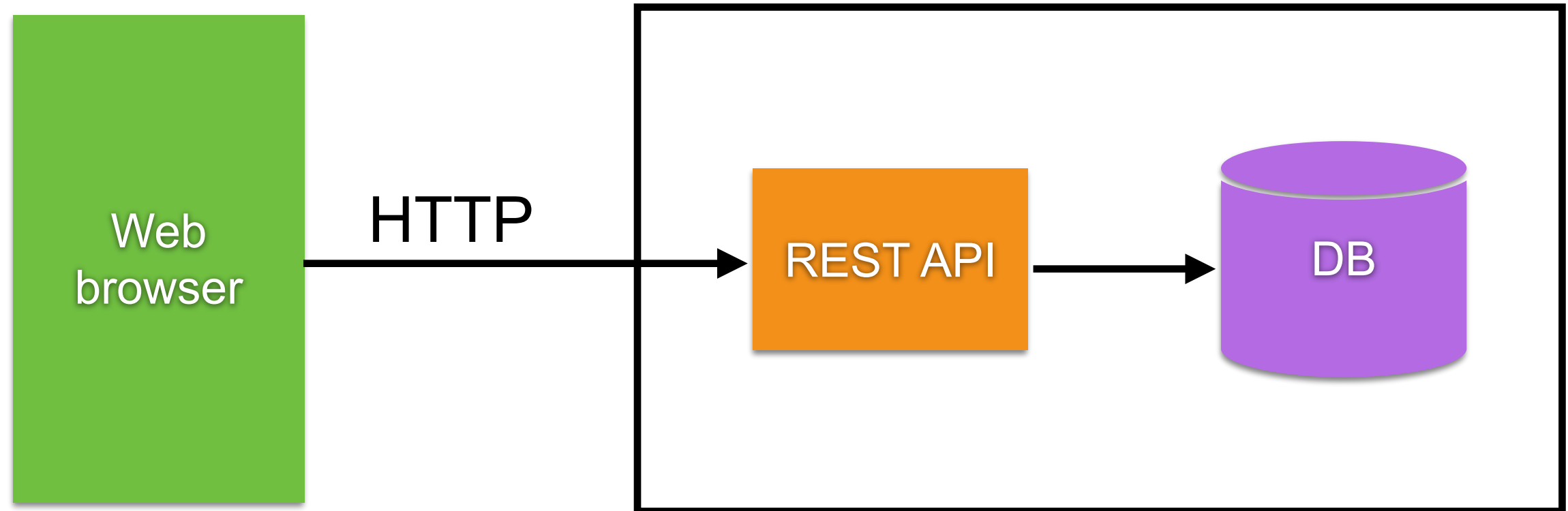
<https://www.cypress.io/>



Web Application

Frontend

Backend

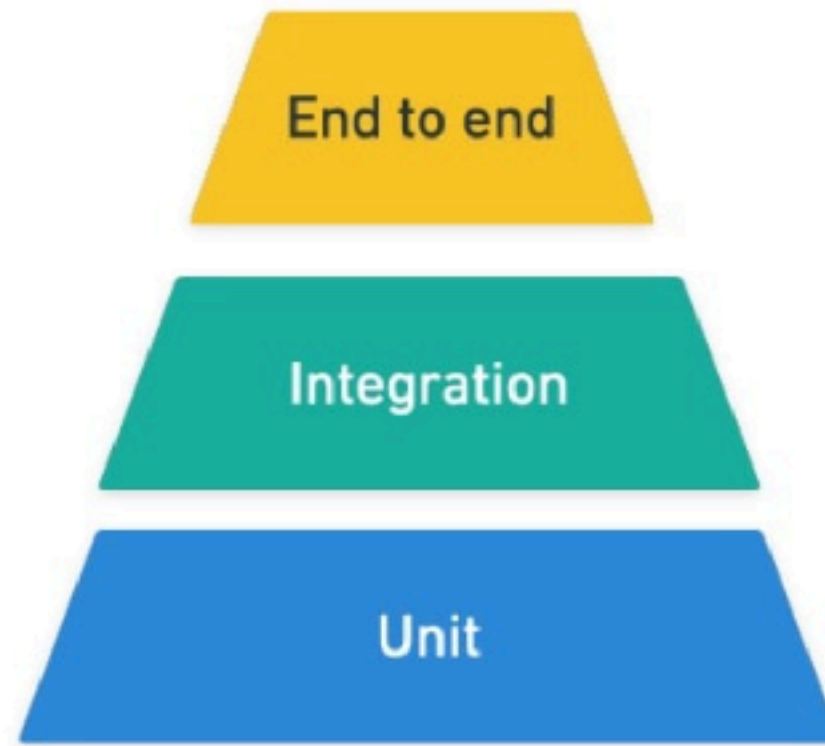


How to test ?



Web Application Testing

- More effort
- Slower
- Higher maintenance



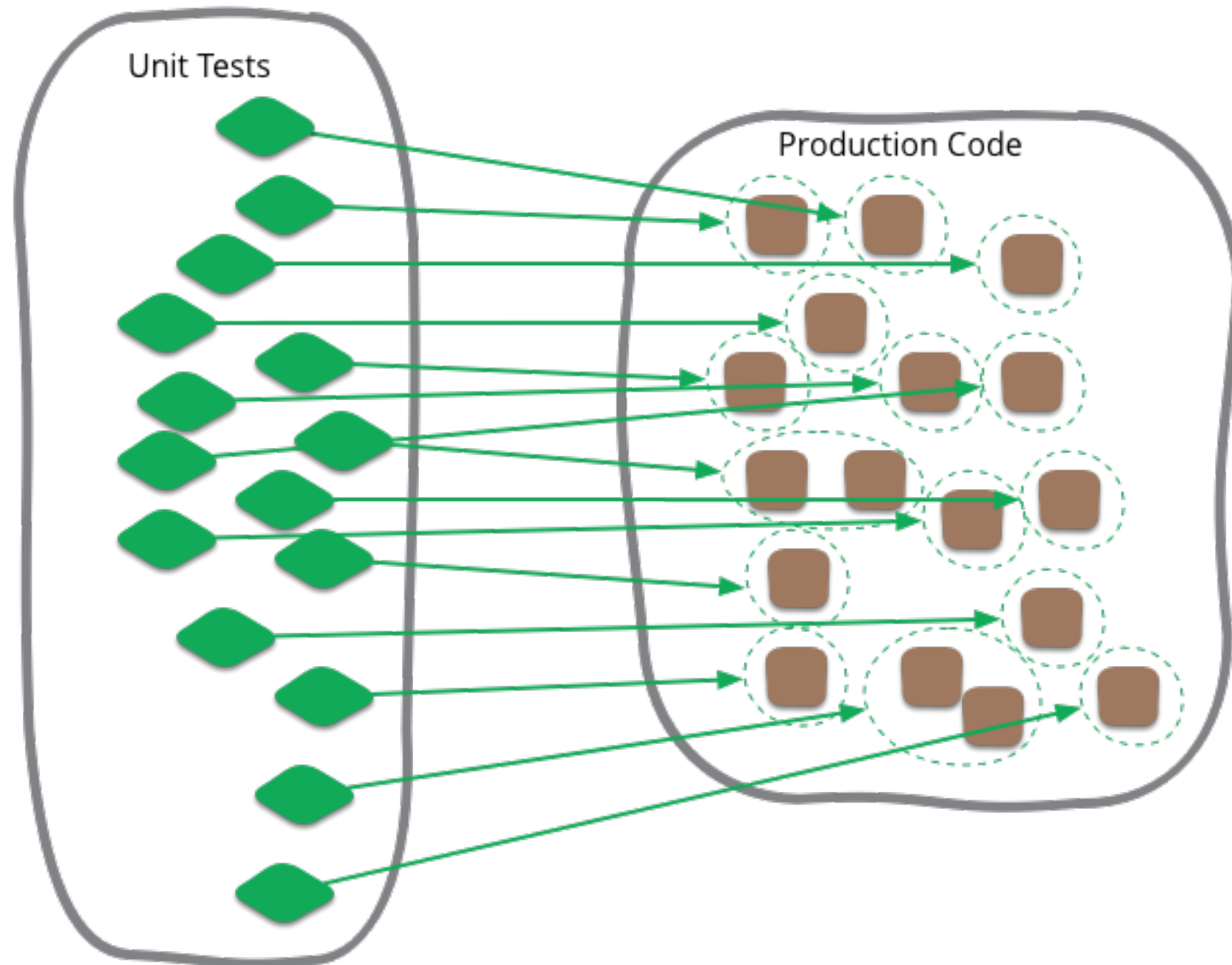
- Faster
- Low effort
- More granularity



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



Unit Tests



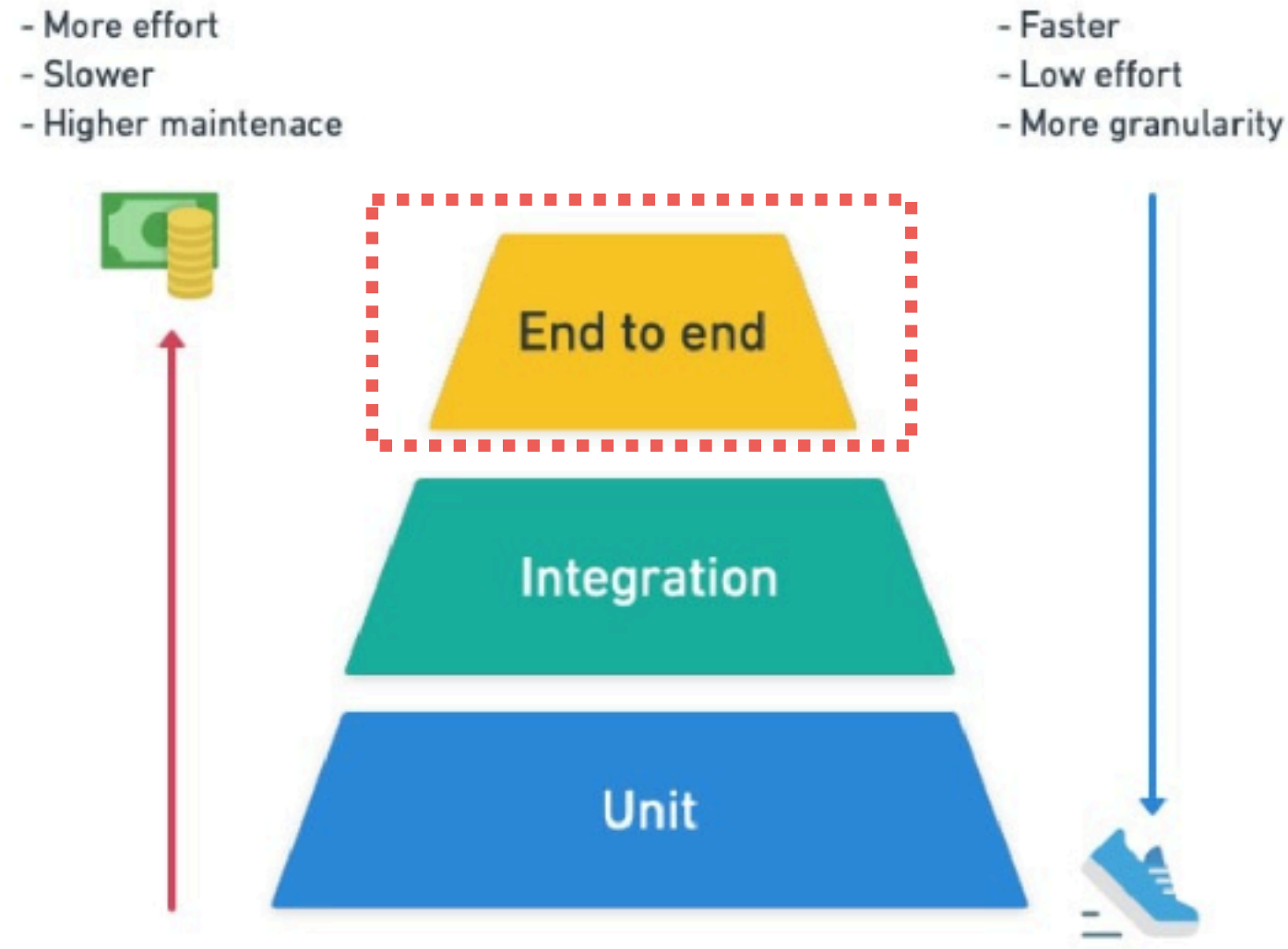
<https://martinfowler.com/bliki/UnitTest.html>



When Unit Tests Passed ...



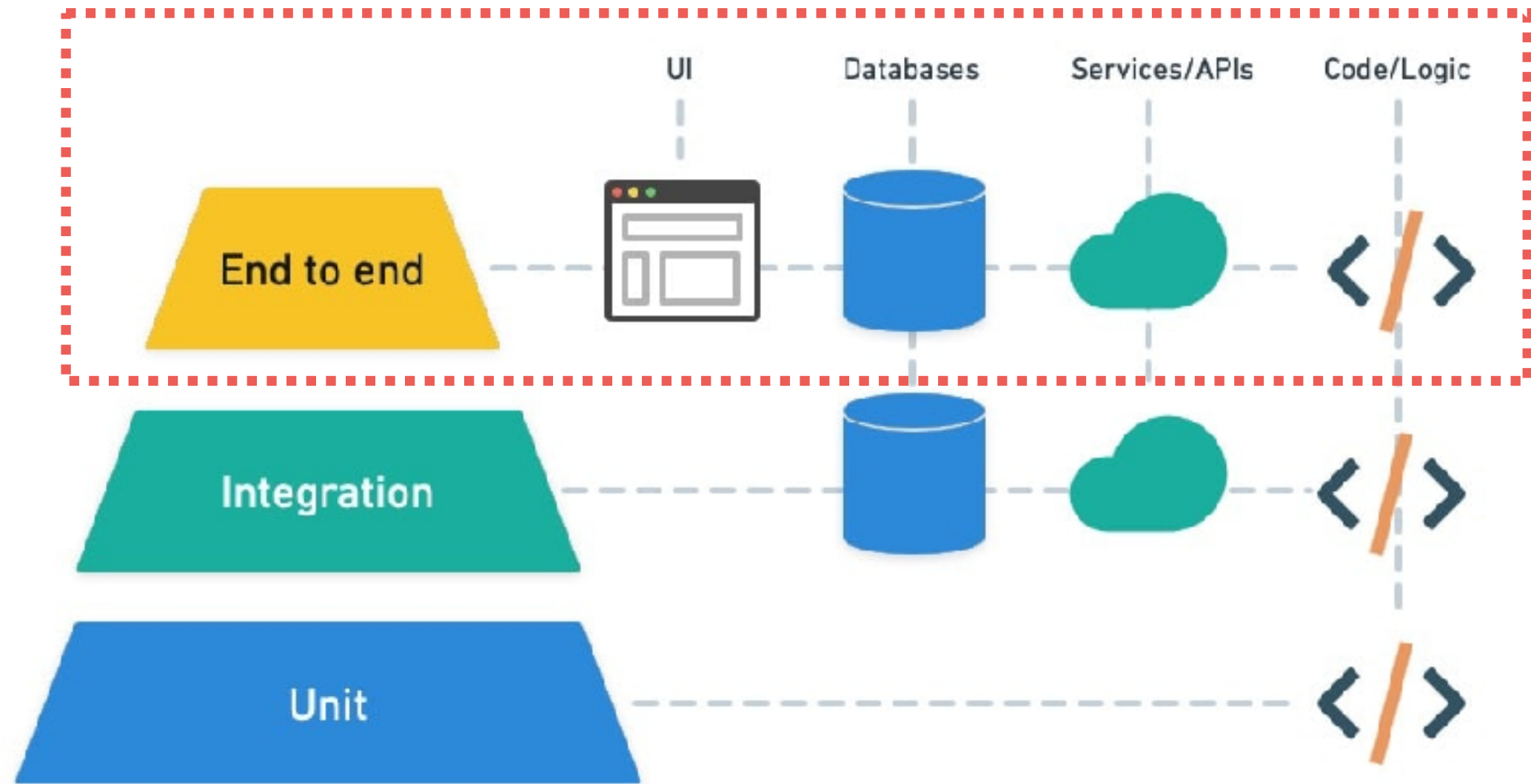
Web Application Testing



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



Web Application Testing



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



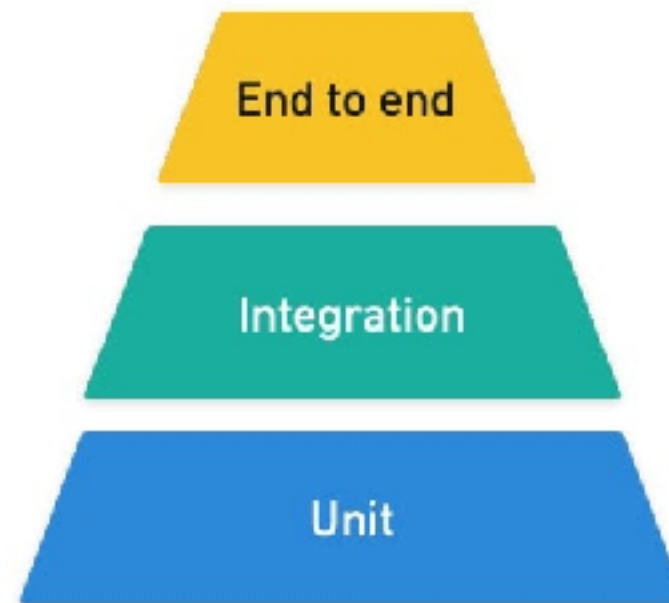
Web Application Testing

Run on real browser

Load read app

Interact with app like a user

The Test Pyramid



History



2005

2010

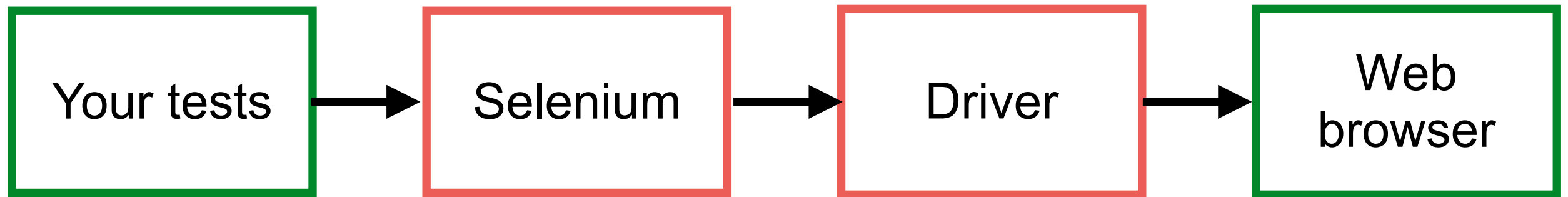
2017

2020

2023



Run on real browser !!



Slow, flaky, unreliable tests



End-to-End Test

Easy to install

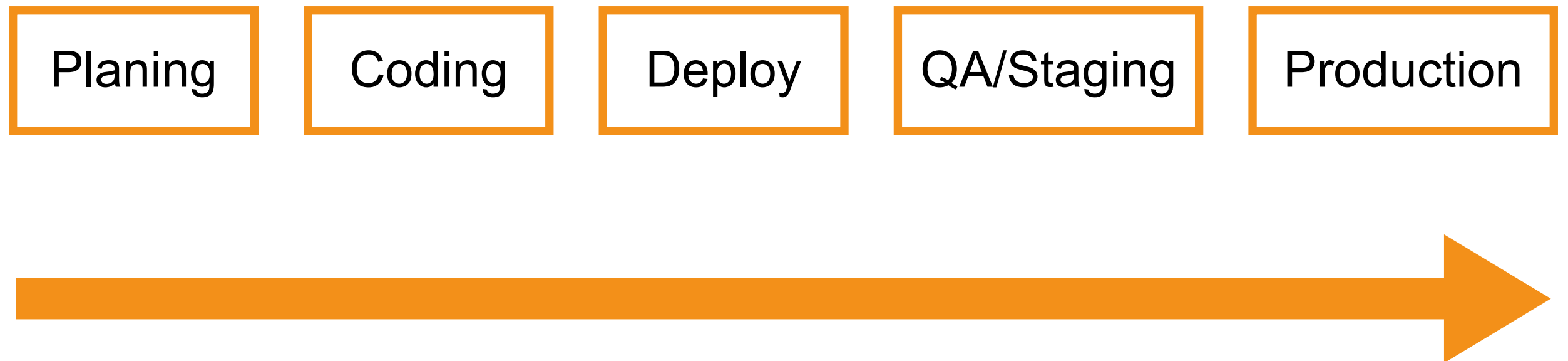
Easy to write tests

Simple to run

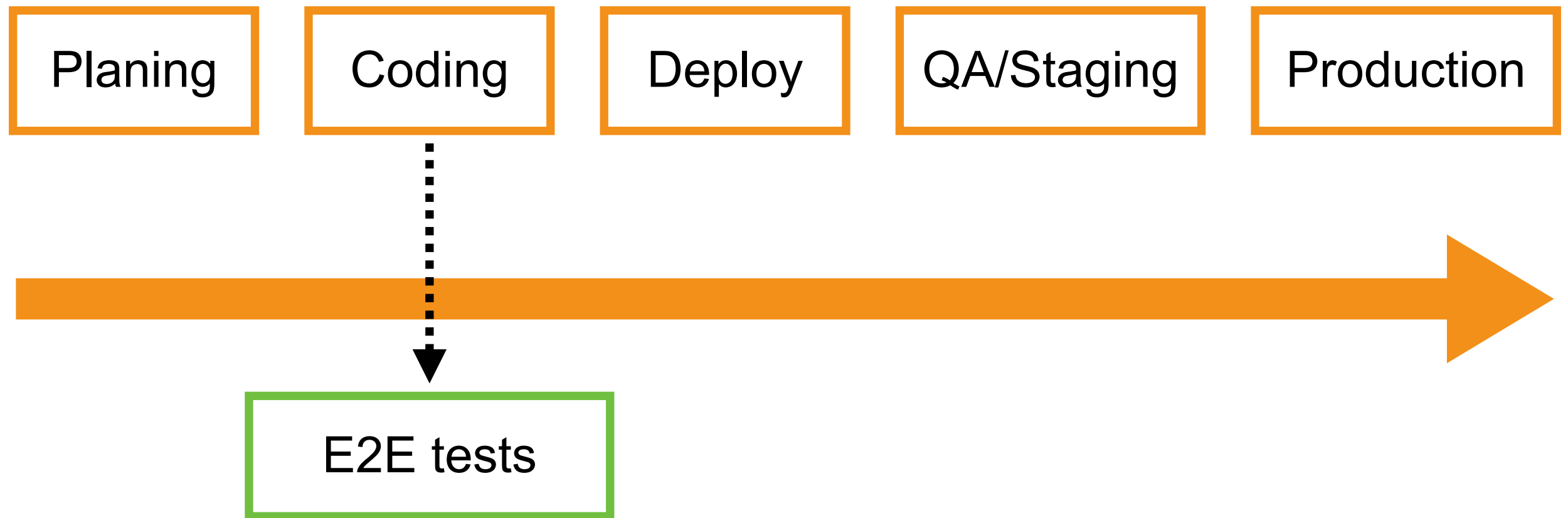
Easy to debug when a test failed



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

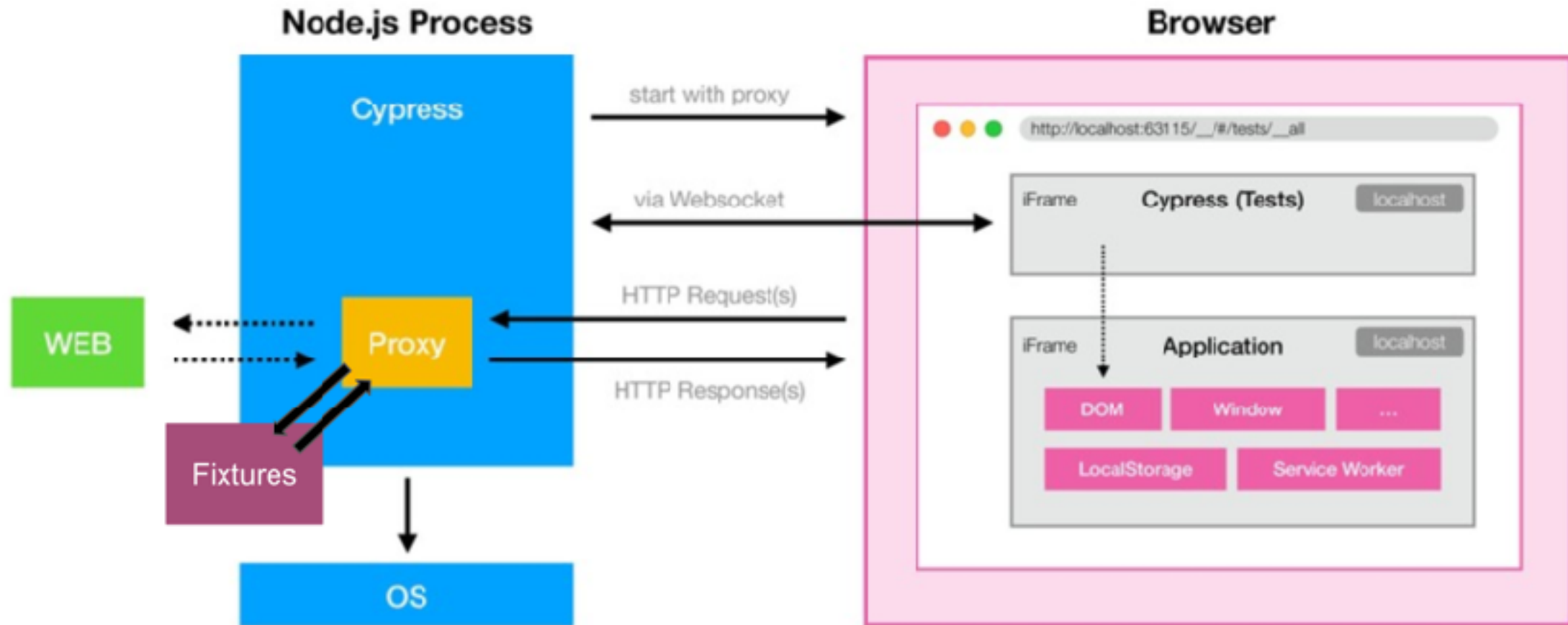
jQuery

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

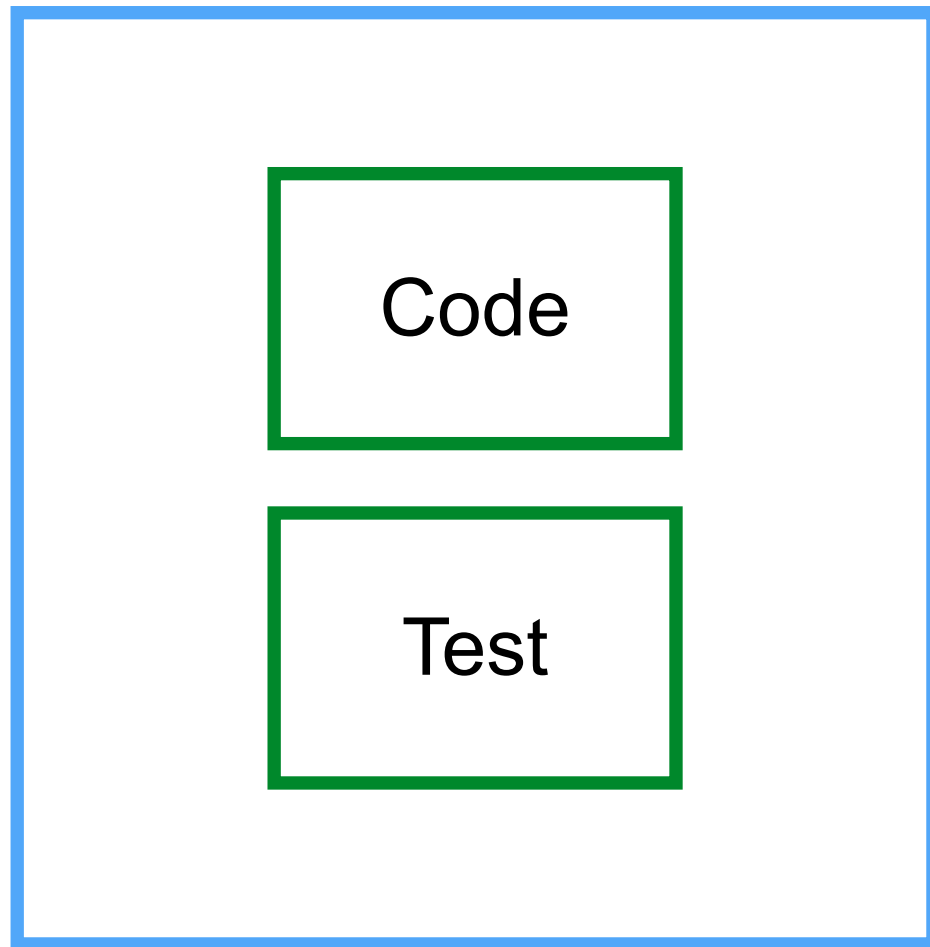


Repository to keep tests ?

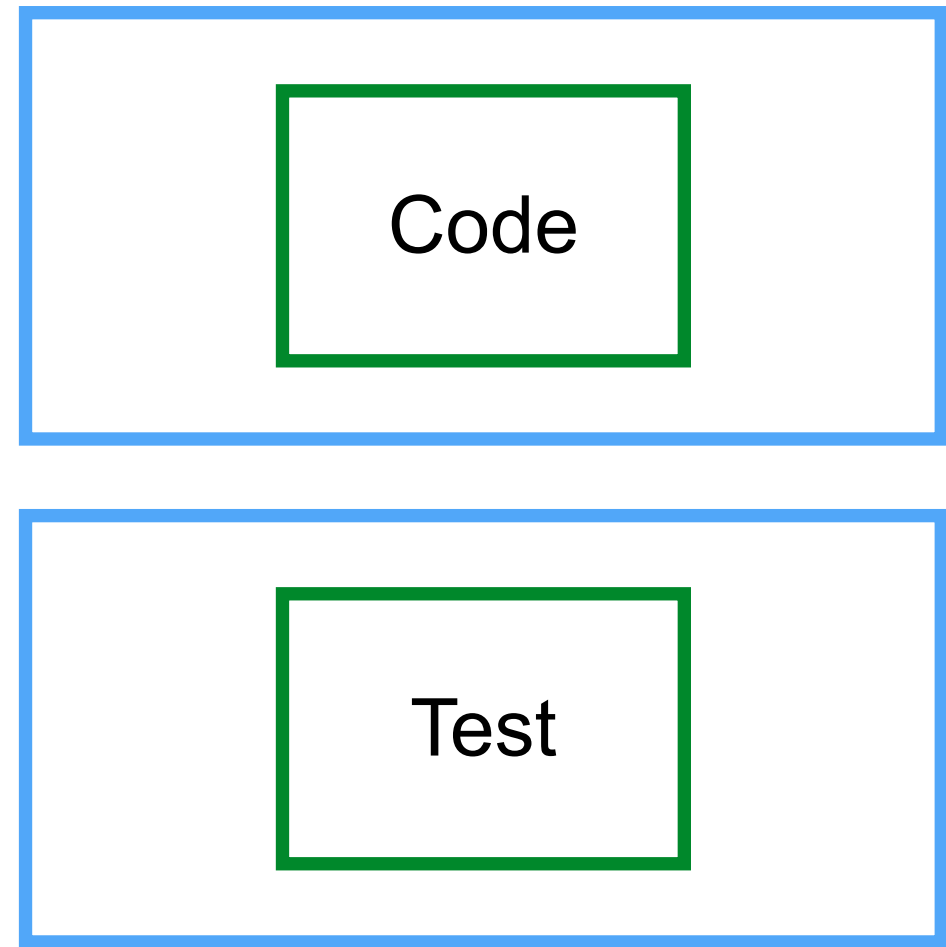


Repository to keep tests ?

Single repository



Multiple repository



Setup for Cypress



Start with Cypress

`$npm init -y`

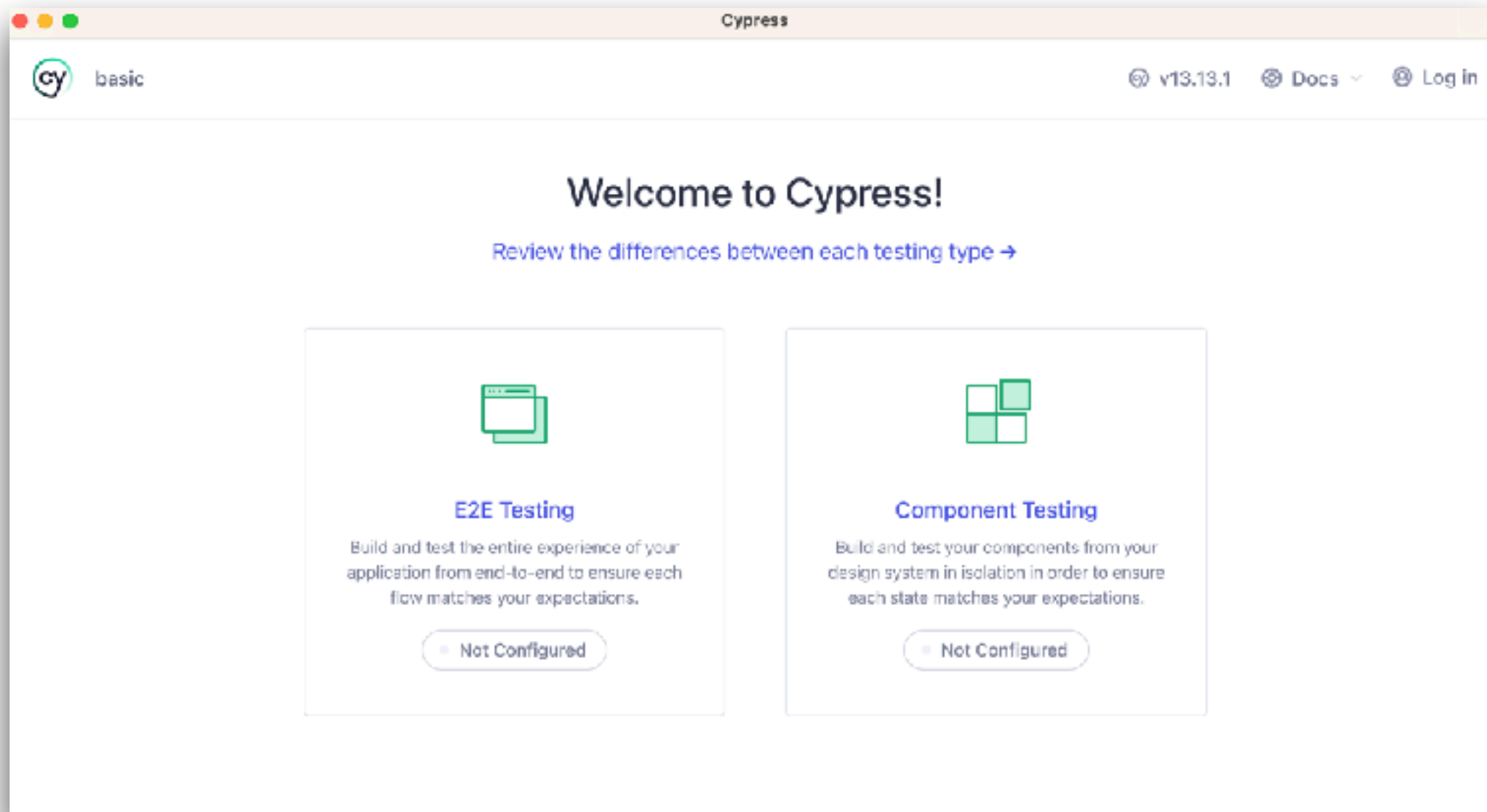
`$npm i -D cypress`

`$npx cypress open`

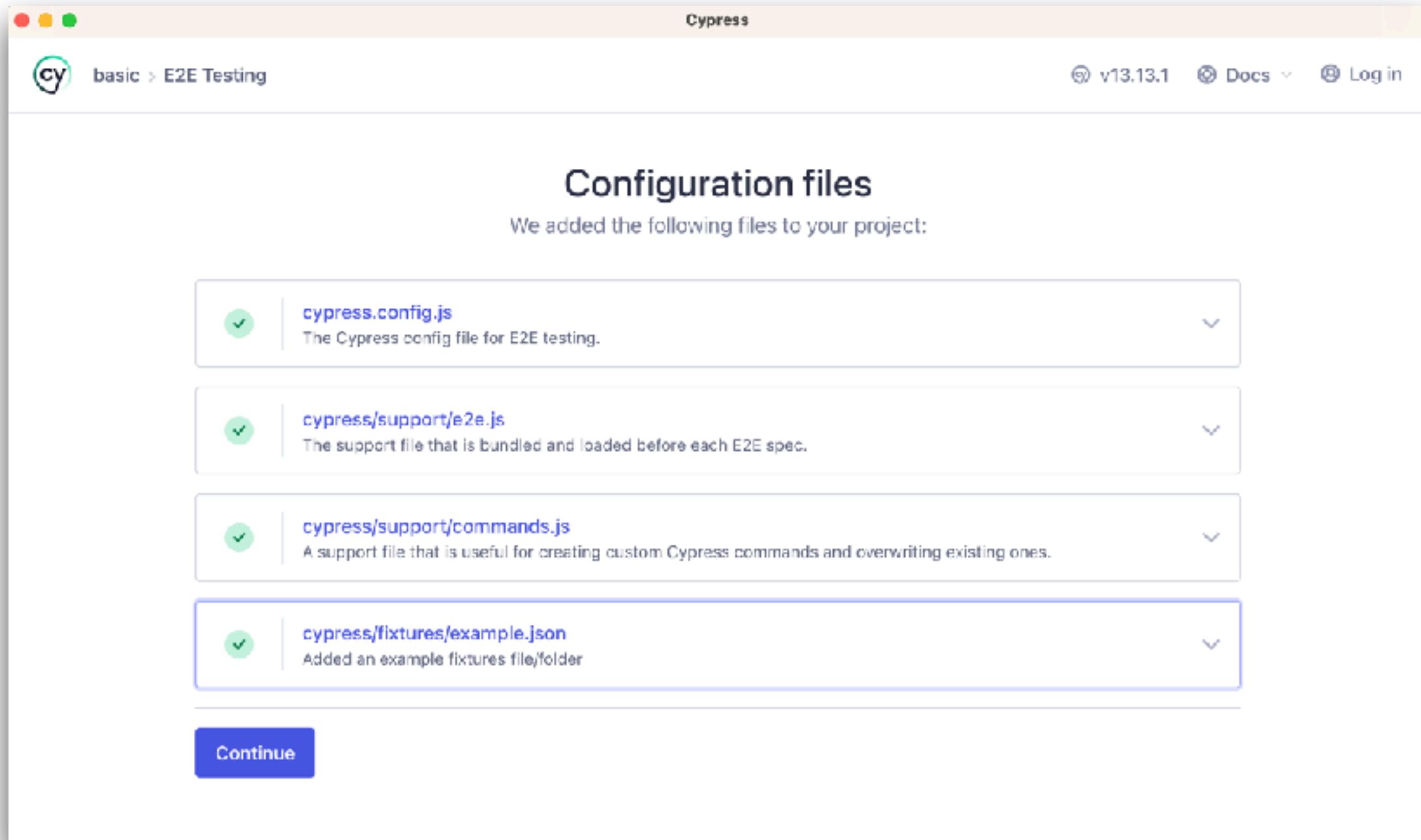


Cypress UI

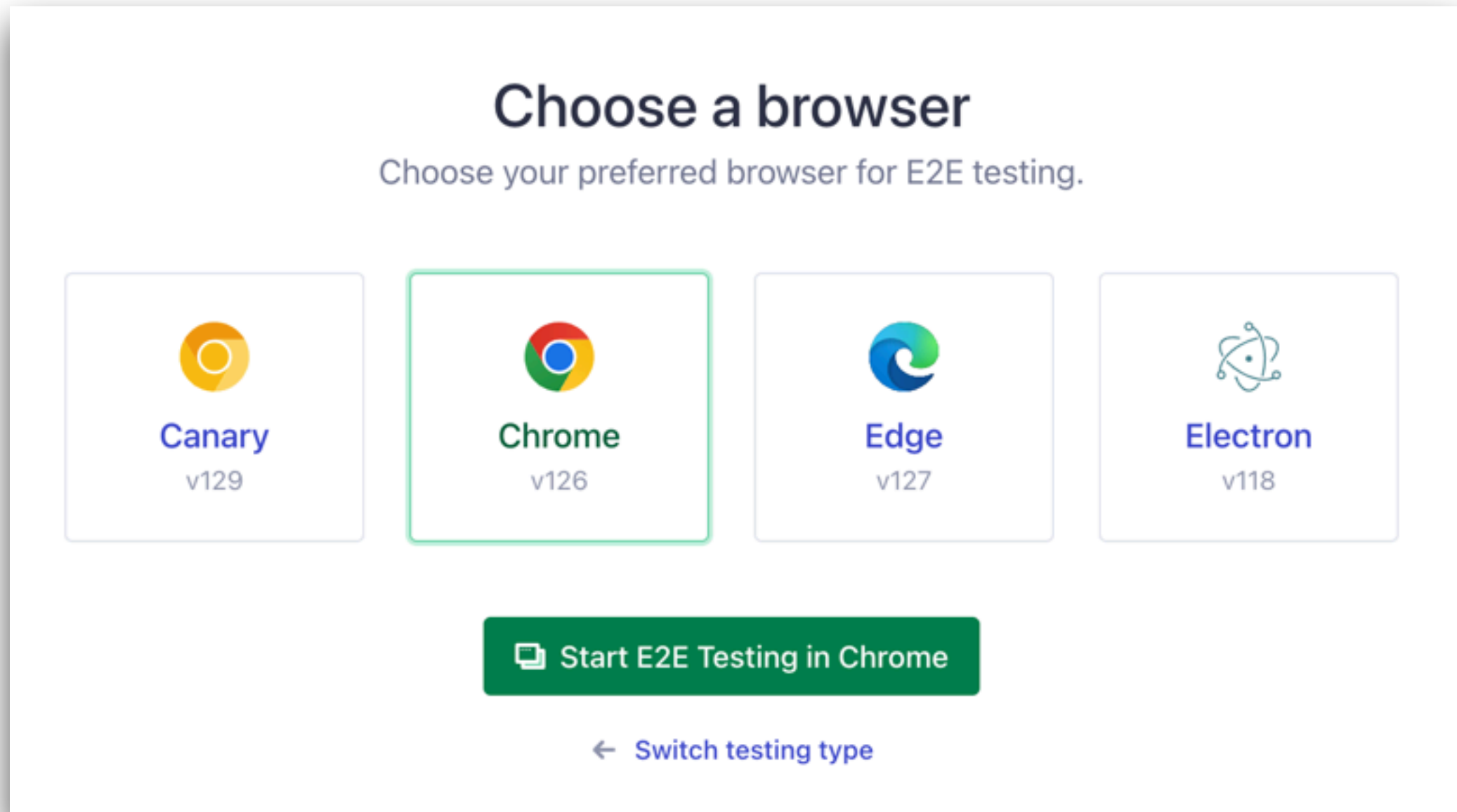
\$npx cypress open



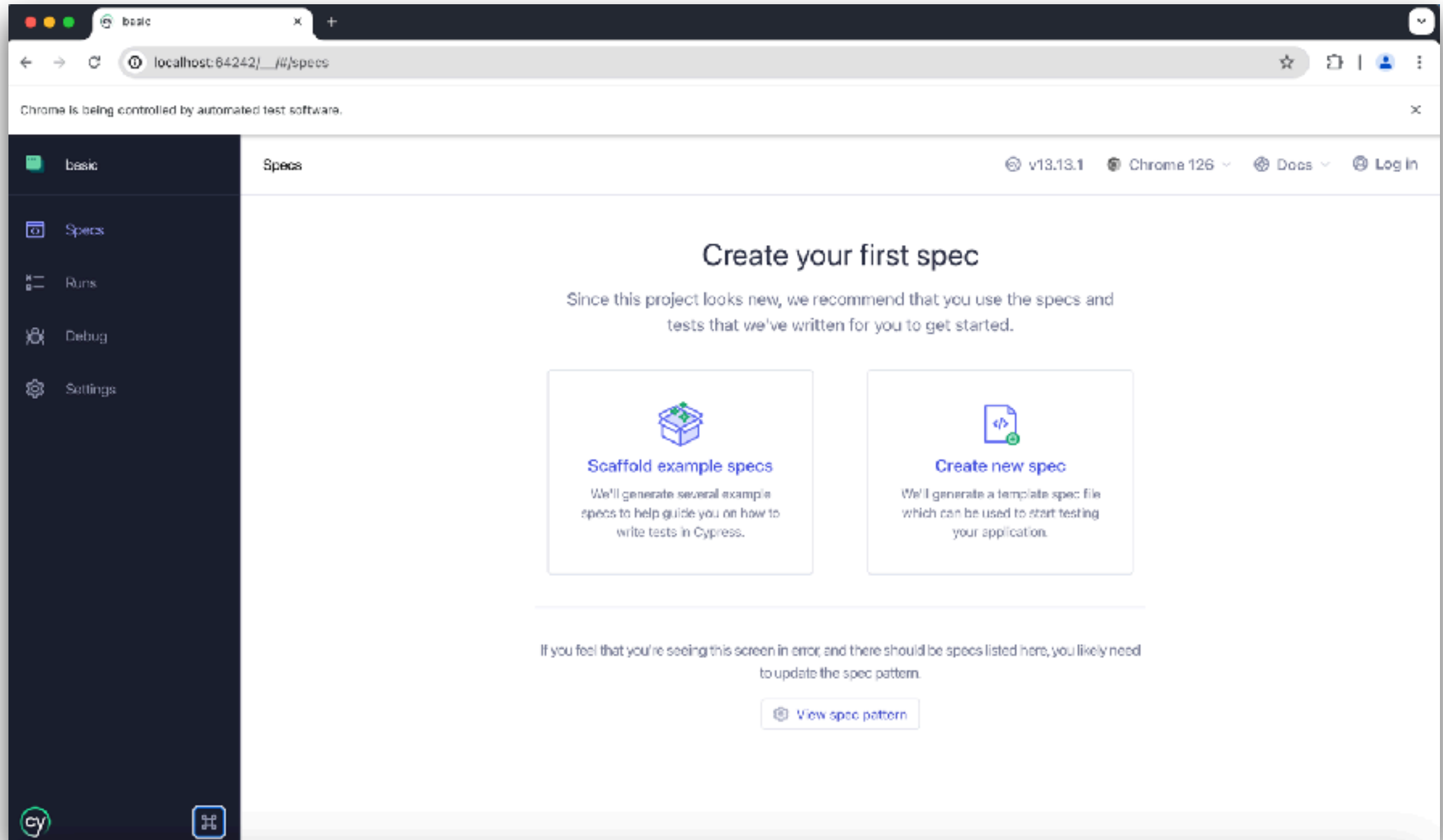
Structure of cypress project



Web browsers for E2E test



Create a first tests/specs



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
screenshot	
video	

<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

Fast

Independent / Isolate

Repeatable

Self-verify

Timely / Thorough

Understandable



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	<code>cy.get('#main').click()</code>
name	<code>cy.get('[name="submission"]').click()</code>
class	<code>cy.get('.btn.btn-large').click()</code>
xPath	Not support
data-cy	<code>cy.get('[data-cy="submit"]').click()</code>

<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 cypress run --headed
Google Chrome	cypress run --browser chrome cypress run --browser chrome --headed
Microsoft Edge	cypress run --browser edge cypress run --browser edge --headed
Firefox	cypress run --browser firefox 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

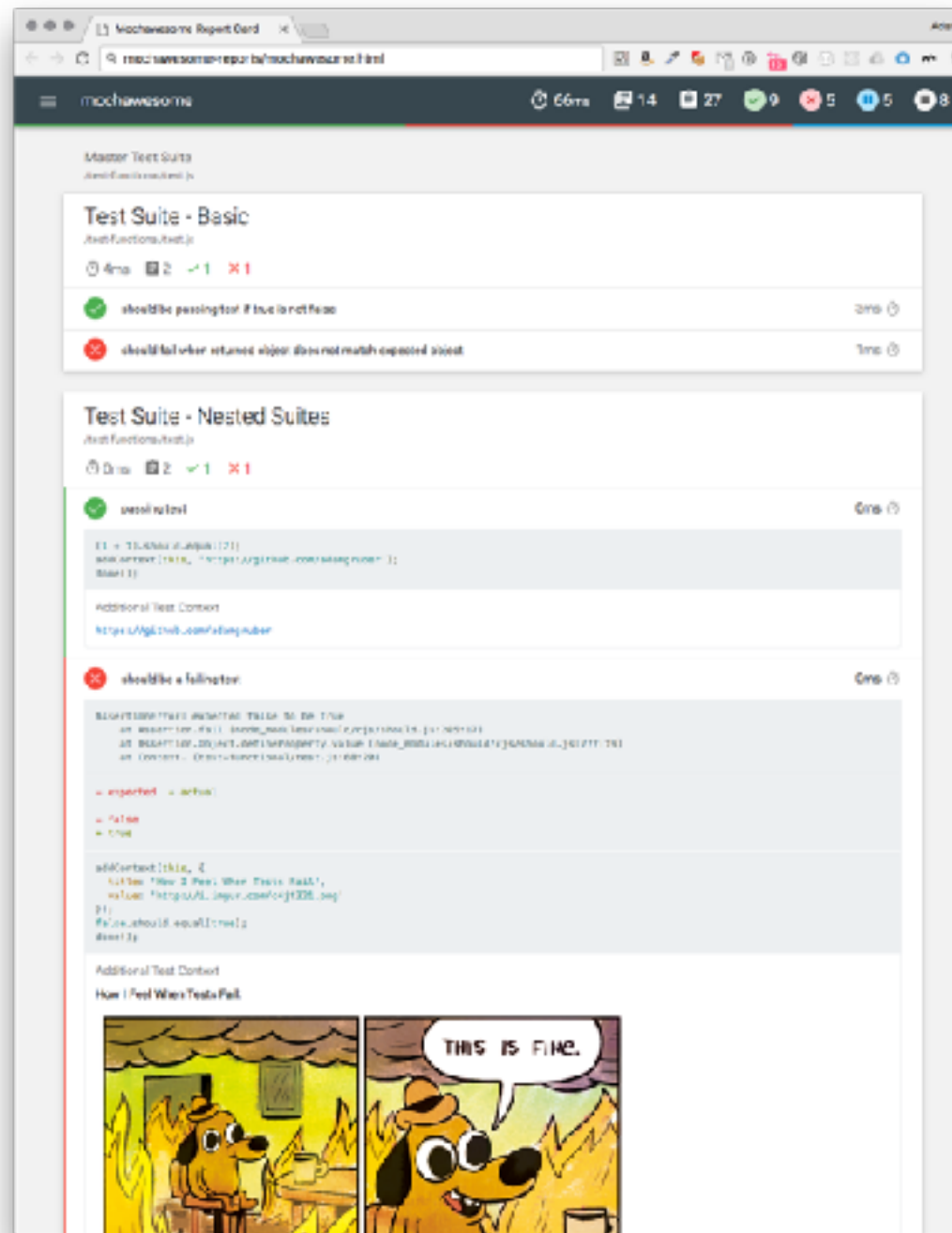
\$npx cypress run --reporter dot -b chrome

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

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



Mocha Awesome



<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

The screenshot shows the Cypress test runner interface. At the top, a dark header bar displays 'basic' on the left and test statistics on the right: a clock icon for '10.1s', a list icon for '1', a clipboard icon for '2', a green checkmark for '1', and a red X for '1'. Below the header, the main area shows a test suite titled 'Search with google.com' with a sub-header '8.8s 2 ✓ 1 ✗ 1'. The test suite contains two tests:

- ✗ Success with search by keyword=cypress** (8.1s): This test failed. The error message is: '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.pZvUc.jf0C1c>' has CSS property: 'display: none''.
- ✓ Success with load first page** (718ms): This test passed. The code snippet shown is:

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



Allure Cypress

Allure

Suites

order name duration status

Status: 0 0 1 0 0 Marks: [pass] [fail] [warning] [info] [retry]

▼ GitHub

▼ GitHub Search

✓ #1 GitHub Search 177ms

Passed GitHub Search

Overview History Retries

Severity: normal

Duration: 177ms

Execution

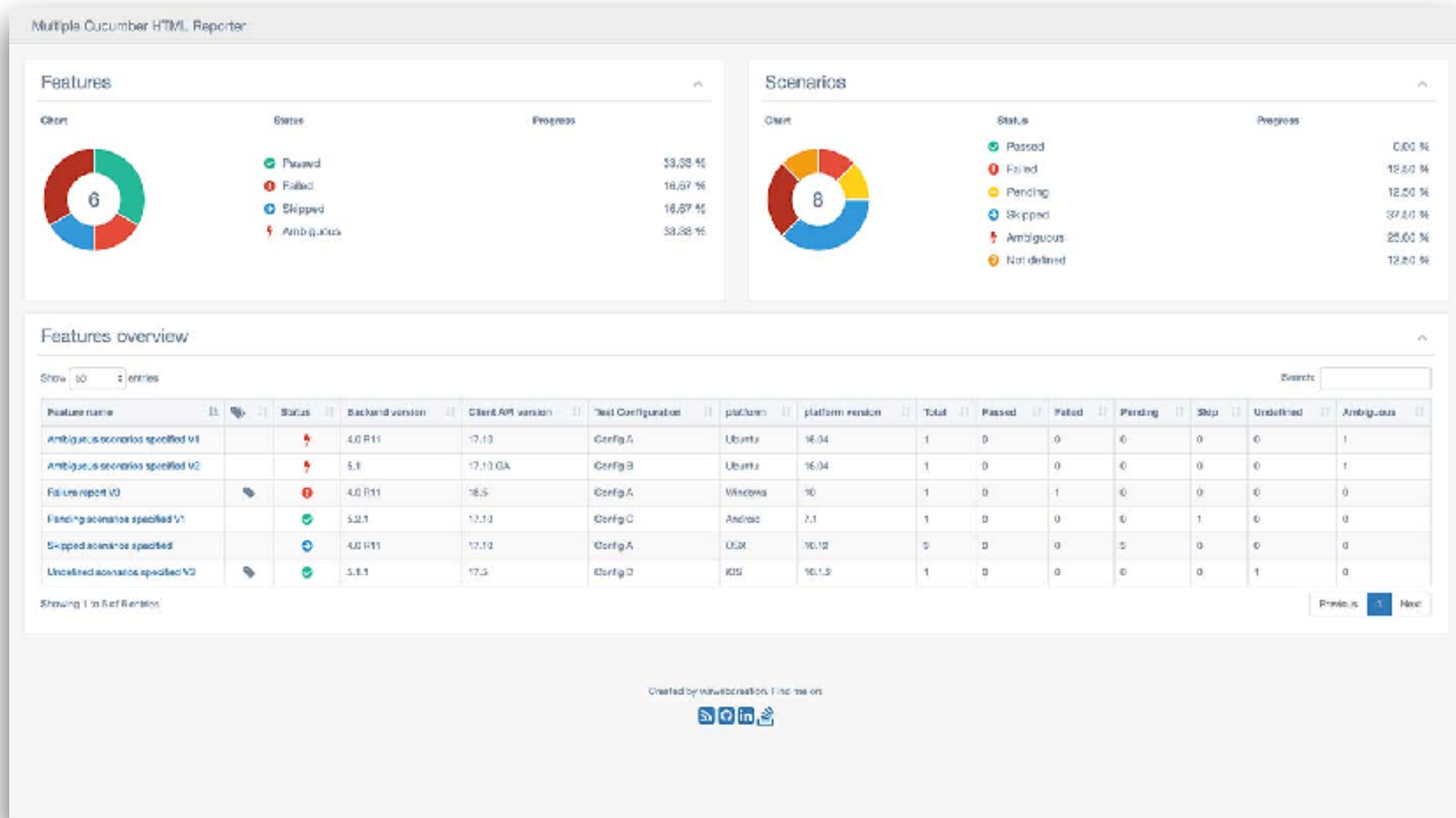
▼ Test body

- ✓ 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 6ms
- ✓ Visit the first search result 4ms

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



Cucumber HTML Report



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



Cypress recording ?



Cypress Studio

Experimental feature

⚠ Experimental

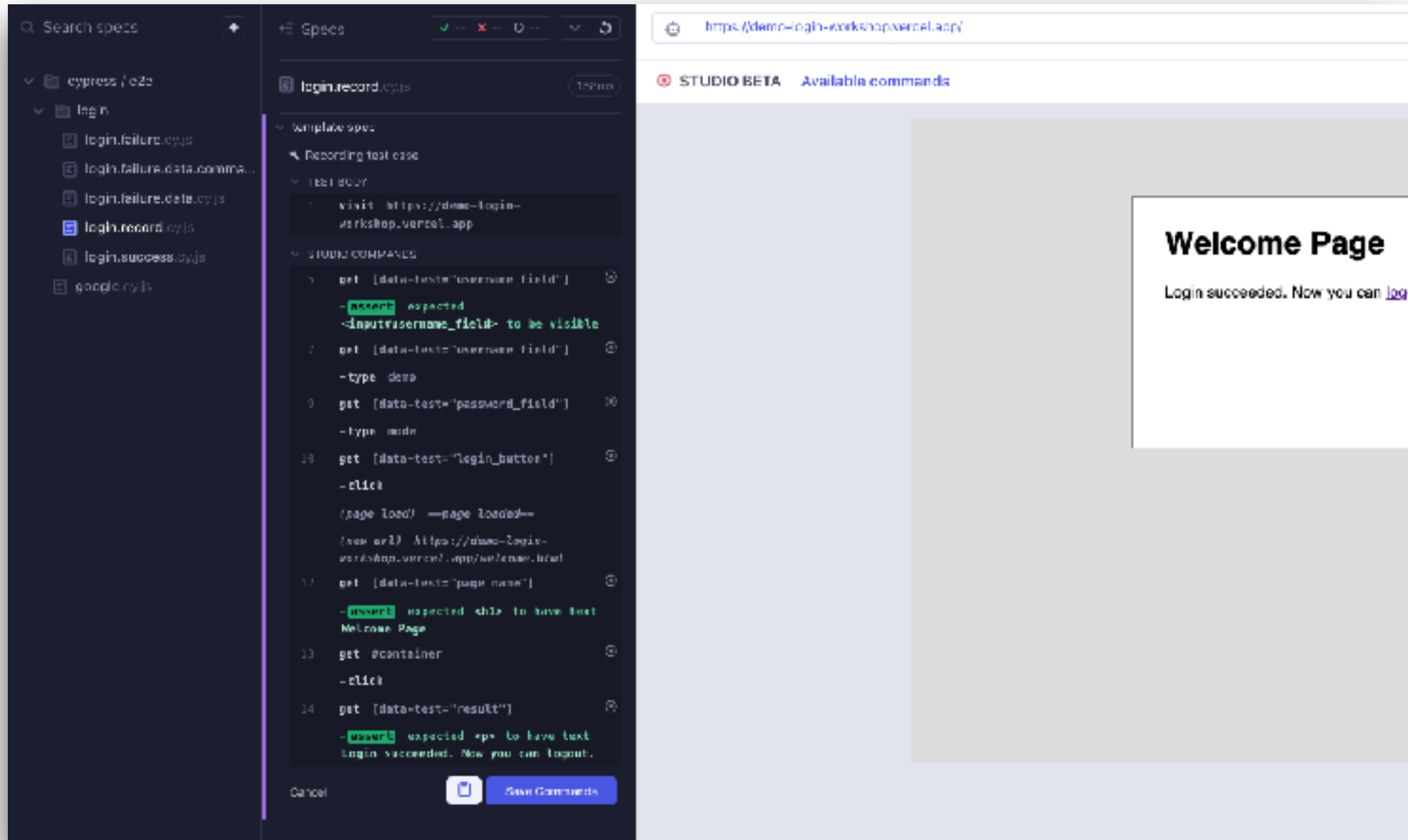
Cypress Studio is an experimental feature and can be enabled by adding the experimentalStudio attribute to your Cypress configuration.

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

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



Restart Cypress UI



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>



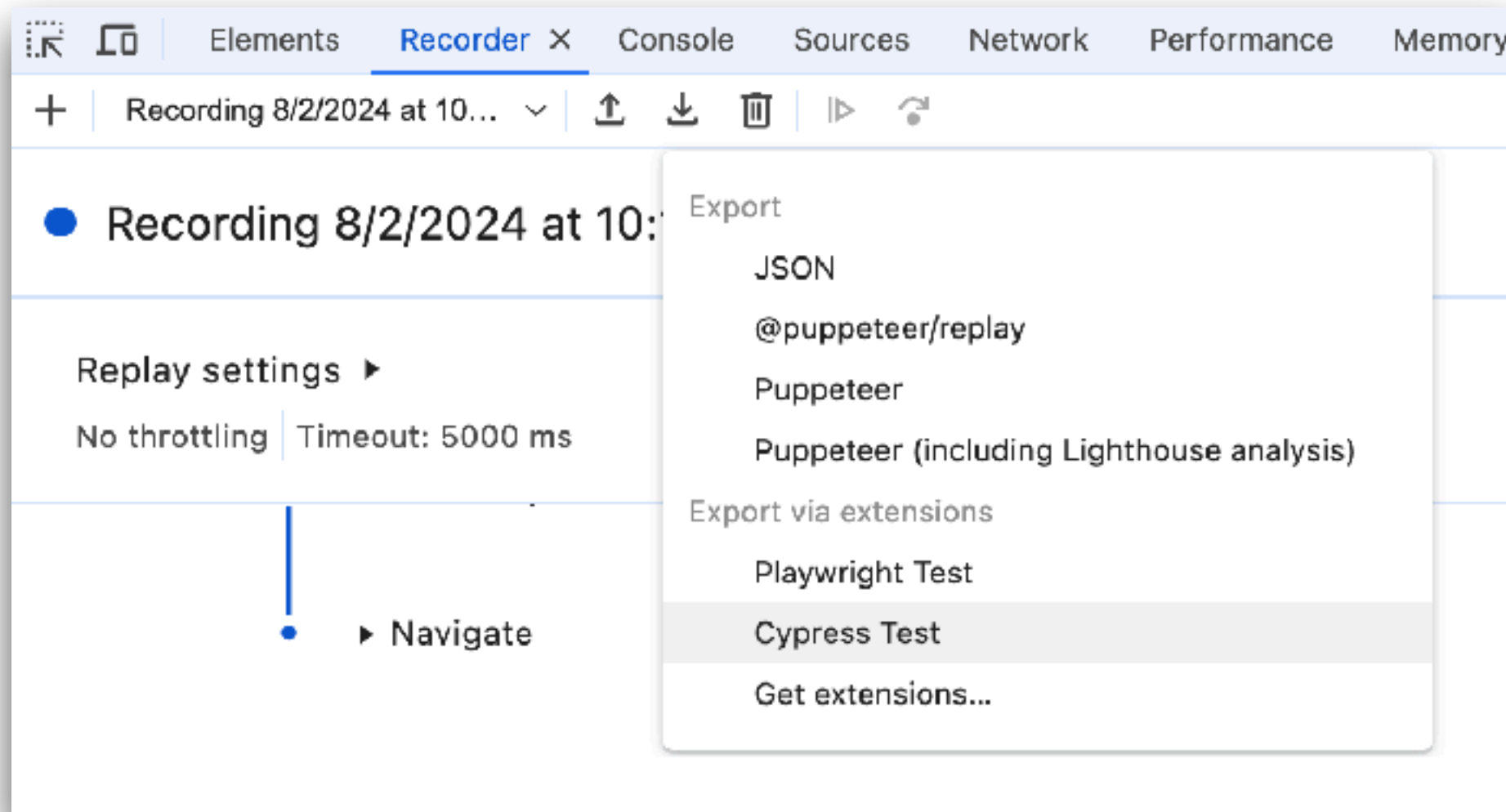
Cypress Chrome Recorder

The screenshot shows the Chrome Web Store page for the 'Cypress Chrome Recorder' extension. At the top, the extension name is displayed with a 'Remove from Chrome' button. Below this, it shows a 'Featured' badge, a 3.0 star rating from 17 reviews, and category tags for 'Extension', 'Developer Tools', and '10,000 users'. The main content area features a video player on the left with the title 'How to use @cypress/chrome-recorder' and a play button. On the right, there's a preview of the extension's interface, showing a tutorial titled 'Installing Cypress and writing your first test' and a sidebar with a list of links, including 'Export as a Cypress test suite' which is circled in red.

<https://chromewebstore.google.com/detail/cypress-chrome-recorder/fellcphjggholofndfmmjmheedhomgin>



Cypress Chrome Recorder



<https://chromewebstore.google.com/detail/cypress-chrome-recorder/fellcphjggholofndfmmjmheedhomgin>



Cypress Recorder

The image shows the Chrome Web Store page for the Cypress Recorder extension. At the top, the extension is titled "Cypress Recorder" with a green icon. It has a 5.0 star rating from 10 reviews, is categorized as an "Extension" and "Developer Tools", and has 10,000 users. A "Remove from Chrome" button is visible. Below this, two screenshots of the extension's interface are shown. The left screenshot shows the initial state with a green header, a large green icon, and the text "Click 'Start Recording' to start recording!". The right screenshot shows the same interface but with a list of recorded Cypress commands in a light blue box, including actions like visiting a URL, clicking elements, and typing text.

Cypress Recorder Remove from Chrome

5.0 ★ (10 ratings)

Extension Developer Tools 10,000 users

Cypress Recorder Info

Click 'Start Recording' to start recording!

```
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



Login-failures.json

List of test datas

```
[  
  {  
    "test_case": "Fail with wrong username",  
    "username": "demo2",  
    "password": "mode"  
  },  
  {  
    "test_case": "Fail with wrong password",  
    "username": "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.username)
      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.username)
      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.username, data.password)  
})
```



Add attribute for test 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



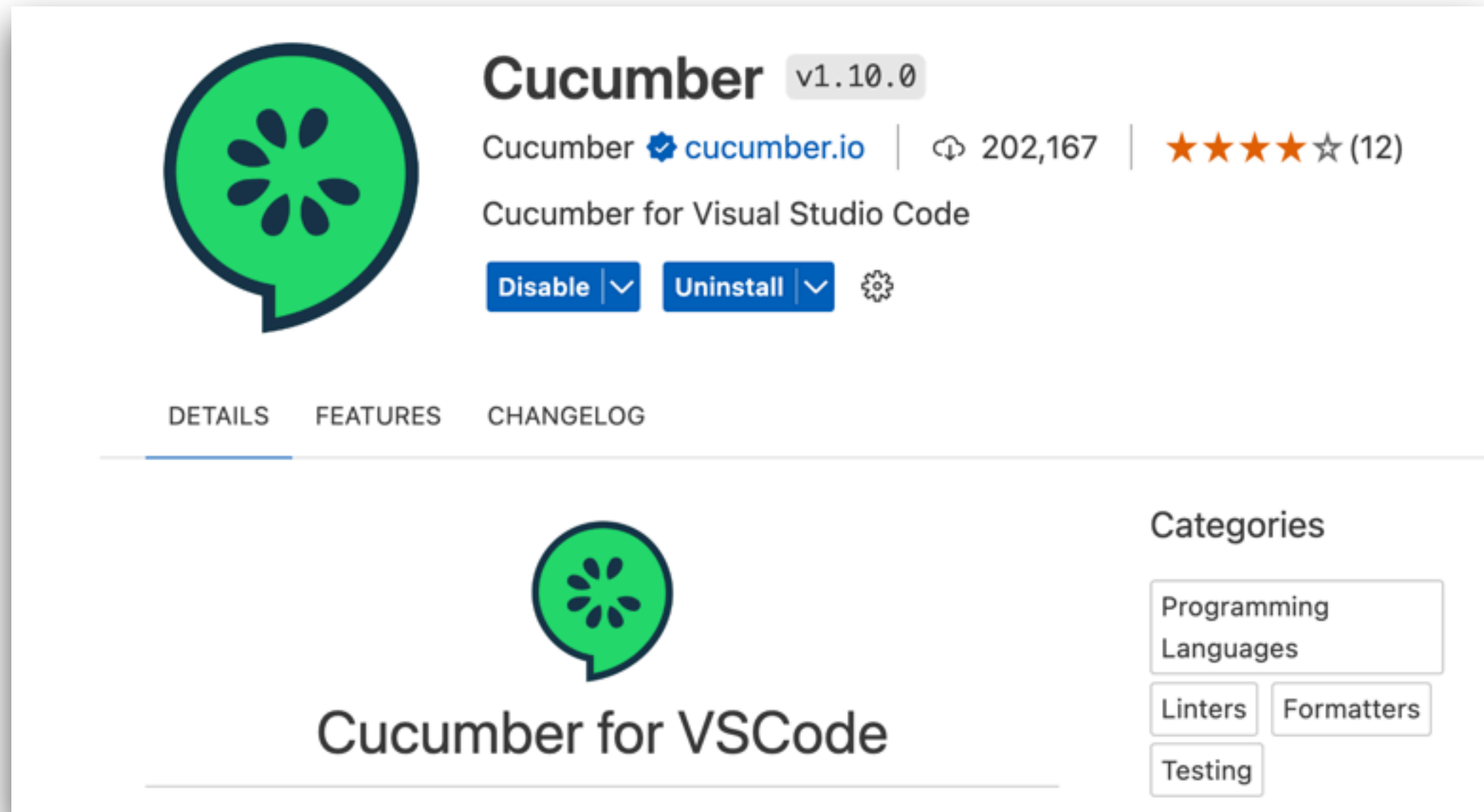
Cucumber

Feature files

Step definition
files





Cucumber plugin in VSCode




The screenshot shows the Cucumber plugin interface in VS Code. On the left is a large green circular icon with a black outline and a stylized black flower-like pattern inside. To the right of the icon, the text "Cucumber" is displayed in a large, bold, black font, followed by a version tag "v1.10.0" in a smaller, grey font. Below this, the text "Cucumber" is followed by a blue checkmark icon and the URL "cucumber.io" in blue. To the right of this is a download icon followed by the number "202,167". Further right are four orange stars and one grey star, followed by the text "(12)". Below this information, the text "Cucumber for Visual Studio Code" is displayed. Underneath are two blue buttons: "Disable" and "Uninstall", each with a small downward arrow. To the right of these buttons is a gear icon. Below the buttons are three tabs: "DETAILS", "FEATURES", and "CHANGELOG". The "DETAILS" tab is selected and underlined. Below the tabs, there is a smaller version of the green circular icon, followed by the text "Cucumber for VSCode". To the right of this, under the heading "Categories", there are three stacked boxes: "Programming Languages", "Linters", and "Formatters".


Cucumber v1.10.0

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

Cucumber for Visual Studio Code

[Disable](#) [Uninstall](#) 

[DETAILS](#) [FEATURES](#) [CHANGELOG](#)



Cucumber for VSCode

Categories

- Programming Languages
- Linters
- Formatters
- Testing



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 !!



Search specs

+ Specs

1 x 0 0

00:02

cypress / e2e / bdd

login.feature

login.feature

Login

Try to login success

TEST BODY

1 When Login with username and password

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

3 get #username_field

4 -assert expected <input#username_field> to be visible

5 -type demo

6 get #password_field

7 -assert expected <input#password_field> to be visible

8 -type mode

9 get #login_button

10 -assert expected <input#login_button> to be visible

11 -click

(page load) --page loaded--

(new url) https://demo-login-workshop.vercel.app/welcome.html

12 Then show welcome page

13 -contains Login succeeded. Now you can logout.

14 -assert expected <p> to be visible

15 get #container > h1

16 -contains Welcome Page

17 -assert expected <h1> to be visible

18 get #container > p

19 -contains Login succeeded. Now you can logout.

20 -assert expected <p> to be visible

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

Electron 118

1000x660 (43%)

DOM Snapshot



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



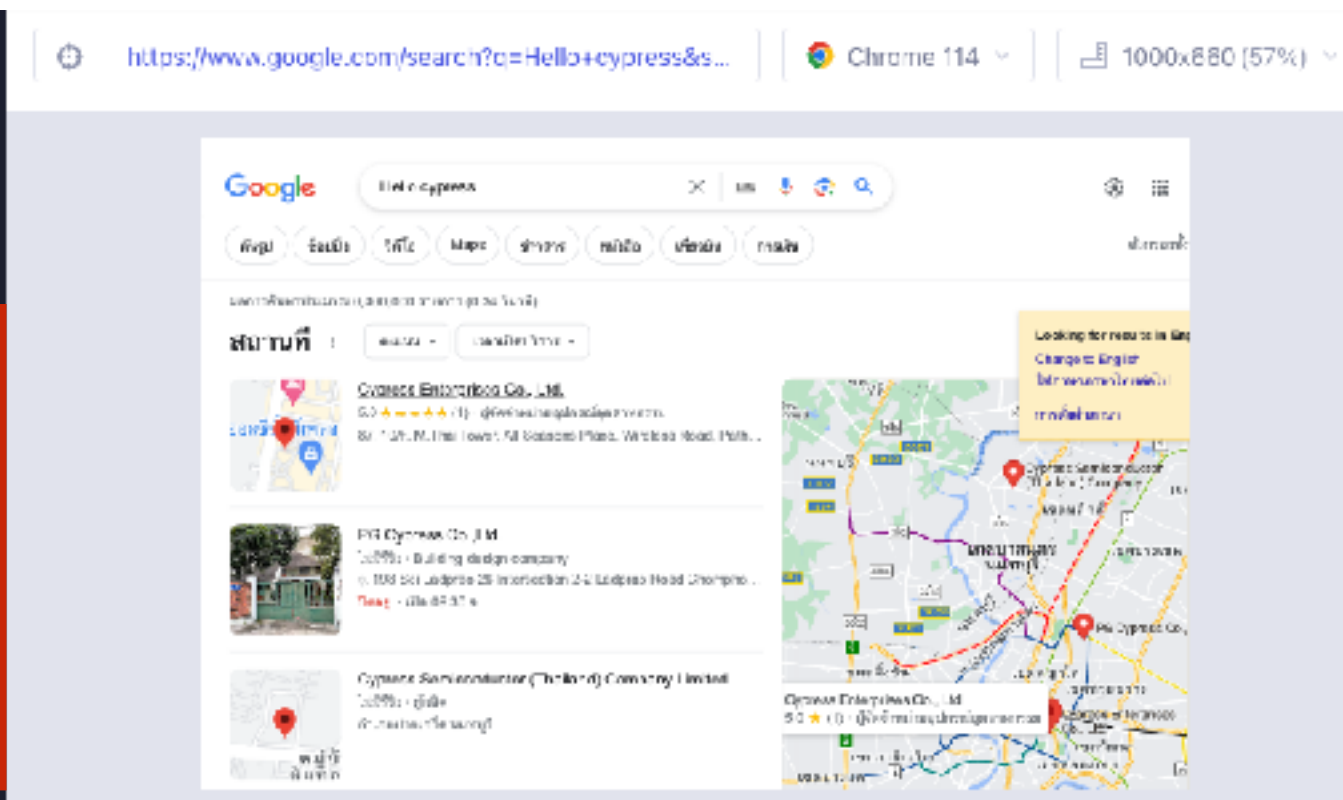
```
Specs
2
google.py,5 00:36
BEFORE EACH
1 visit https://www.google.com

(xhr) GET 200 /complete/search?
q&cp=86client=gws-
wiz&ssi=t&gs_l=th&authuser=86psi=u66FZP
bvC47e1sQP7p1D1AE.16864826281846dpr=16molsbt=1

(xhr) GET
/xjs/_/js/md=1/k=xjs.s.th.U_t0TRfb7oQ.0/an=CAIAI
AAg5oK7ABtAgAAAAQAAICBAAAAAAGAEYAAgeJQRAAAAIcFpE
MQBAAwAJTQAAAAADj9EAAAAAE0BAAAGeIAMGgIKAA CAIAI
N8ADHgBAAYTFgAAAAAIAAEH15BIMbJEB8AAgAAAAAIAA
FTJ5MWB EA/rs=ACT90cN8enprgo0hL3aQYwV968zD1pJ0

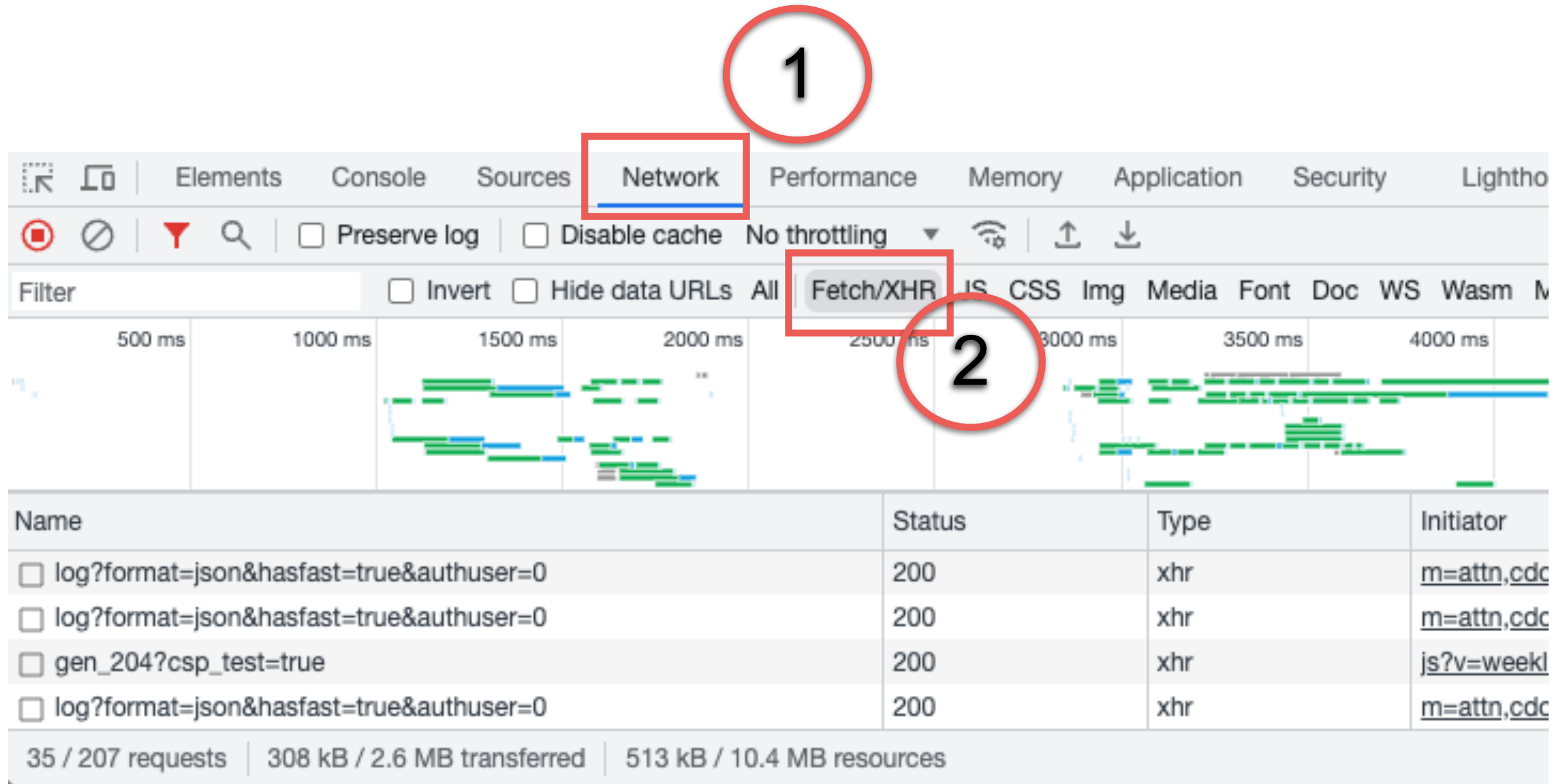
(xhr) POST 200 https://play.google.com/log?
format=json&hasfast=true&authuser=8

(xhr) GET 204 /client_204?cs=1&api=89978449
```



XHR in Google Chrome

1



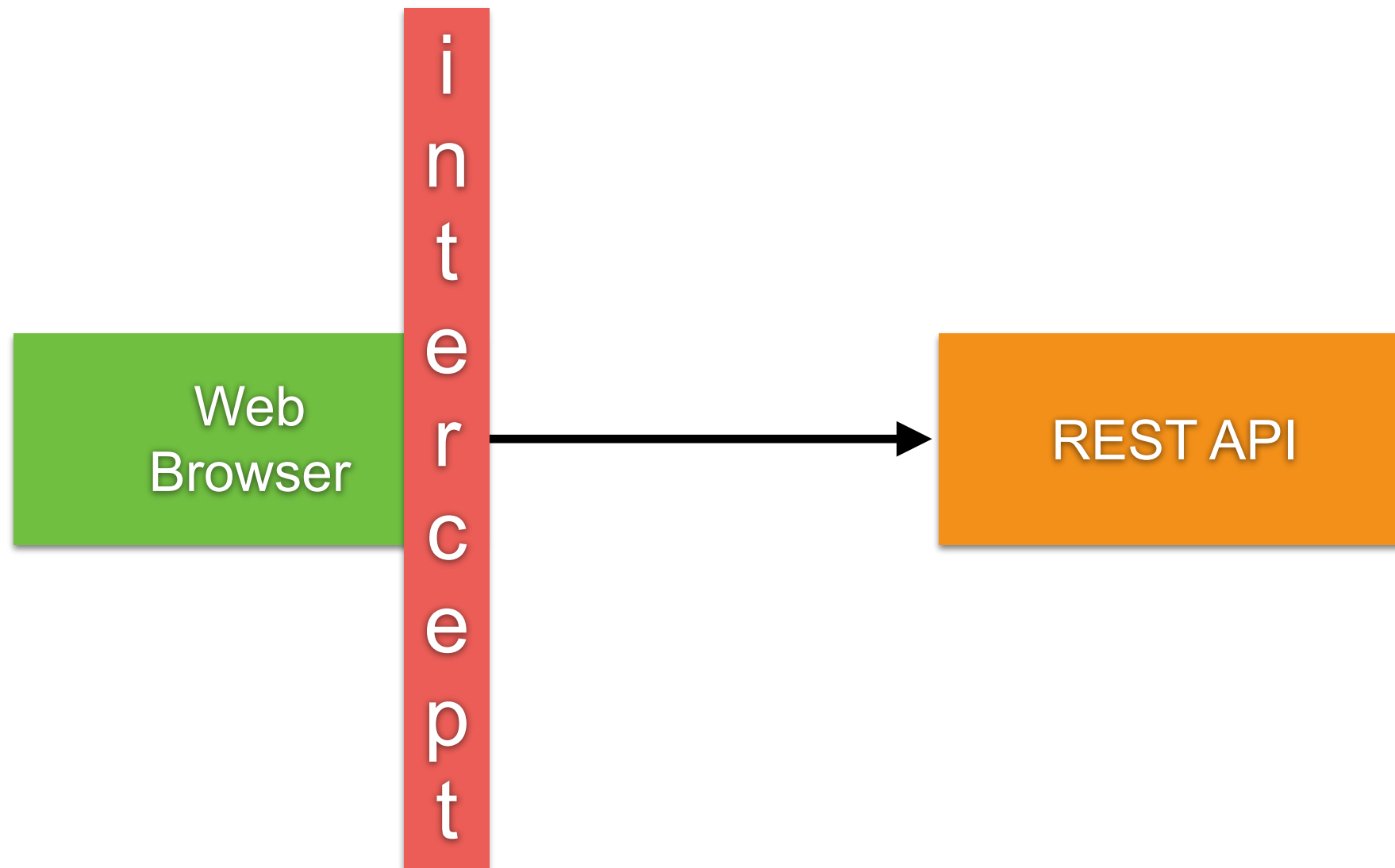
Name	Status	Type	Initiator
<input type="checkbox"/> log?format=json&hasfast=true&authuser=0	200	xhr	m=attn,cdd
<input type="checkbox"/> log?format=json&hasfast=true&authuser=0	200	xhr	m=attn,cdd
<input type="checkbox"/> gen_204?csp_test=true	200	xhr	js?v=weekl
<input type="checkbox"/> log?format=json&hasfast=true&authuser=0	200	xhr	m=attn,cdd

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

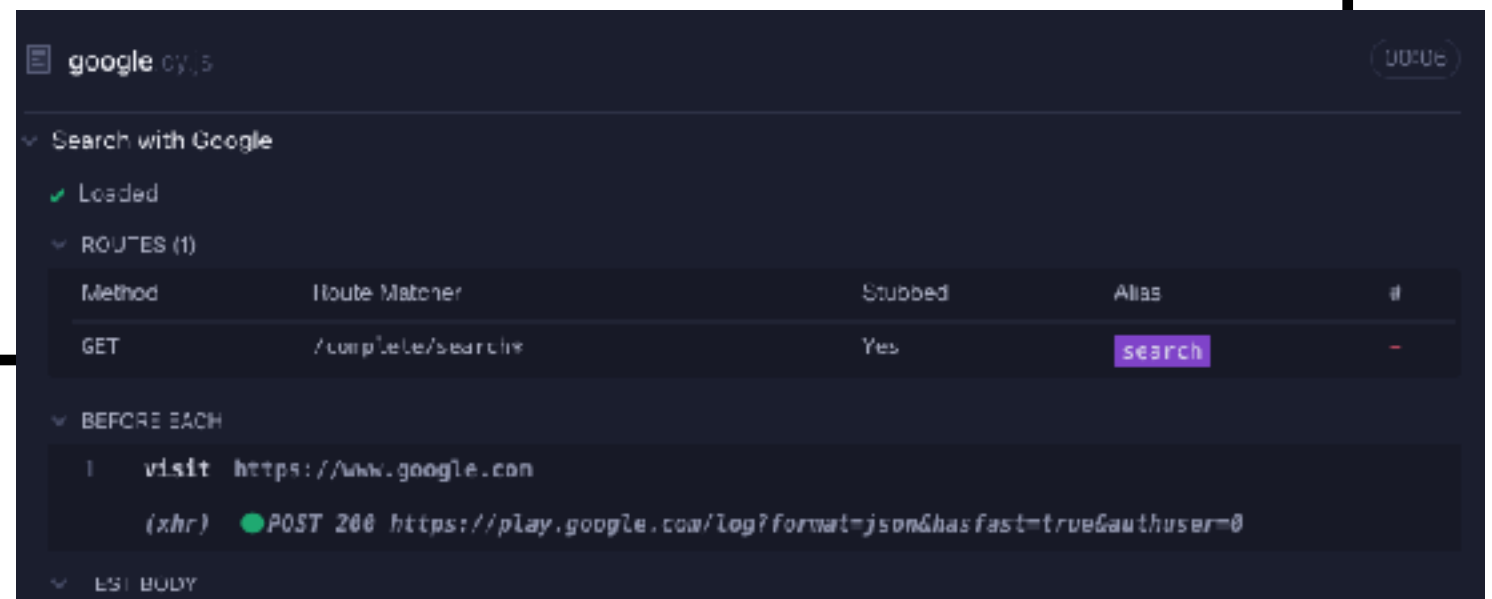
<https://docs.cypress.io/guides/guides/network-requests>



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')  
  })  
})
```



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

```
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

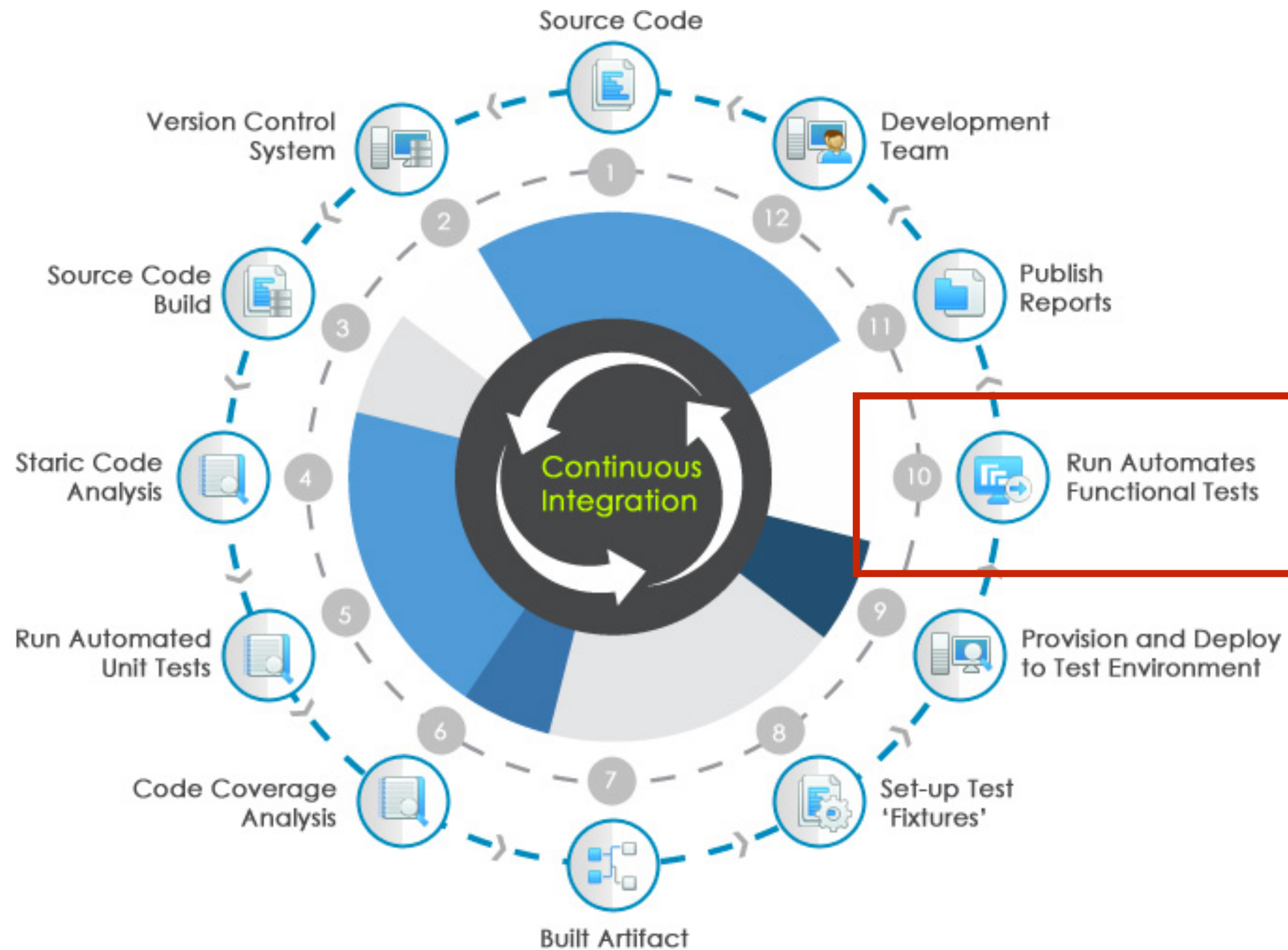
```
services:
  testing:
    build:
      context: .
      dockerfile: Dockerfile
```



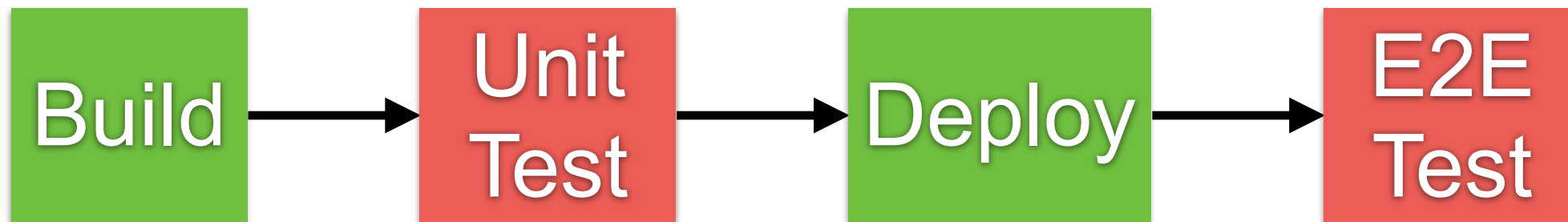
Continuous Integration



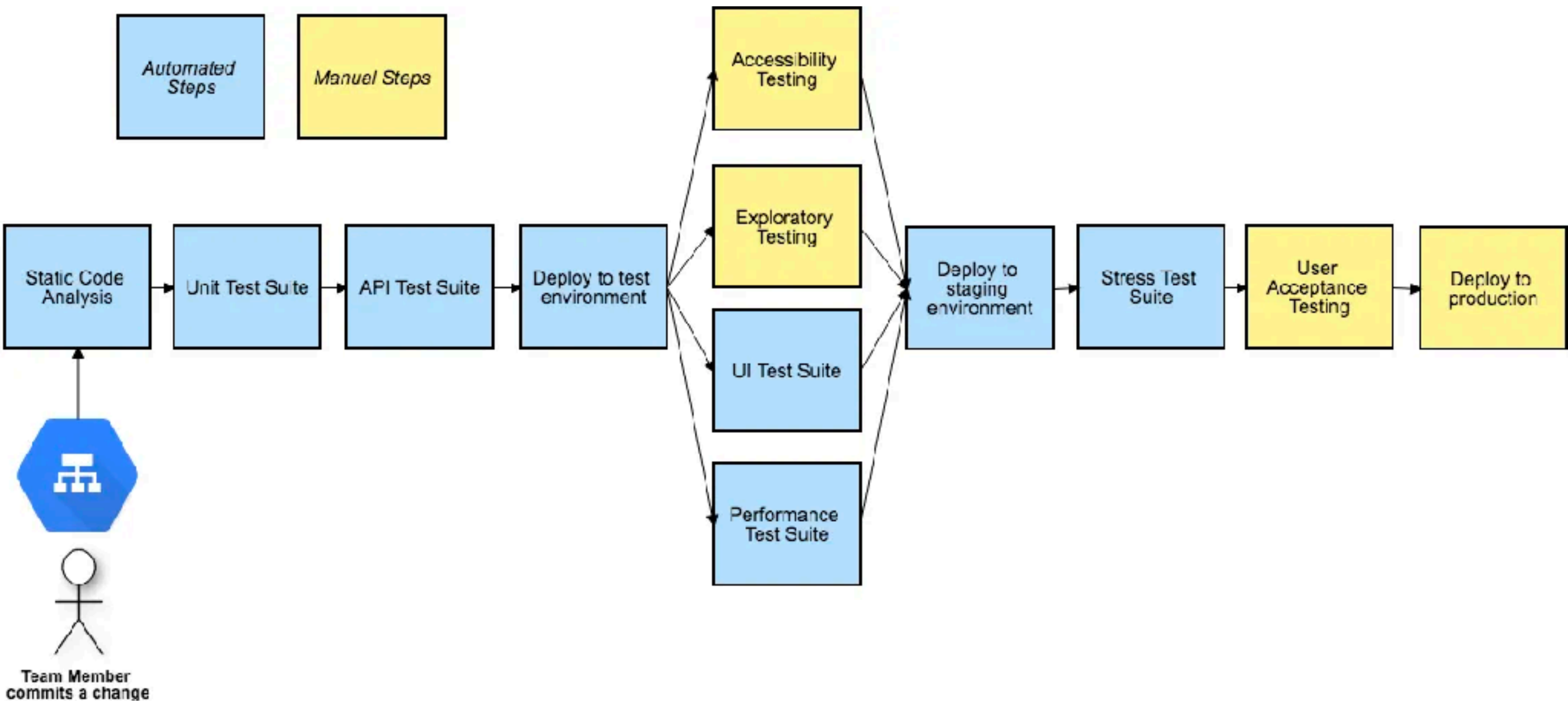
Continuous integration



Design your pipeline



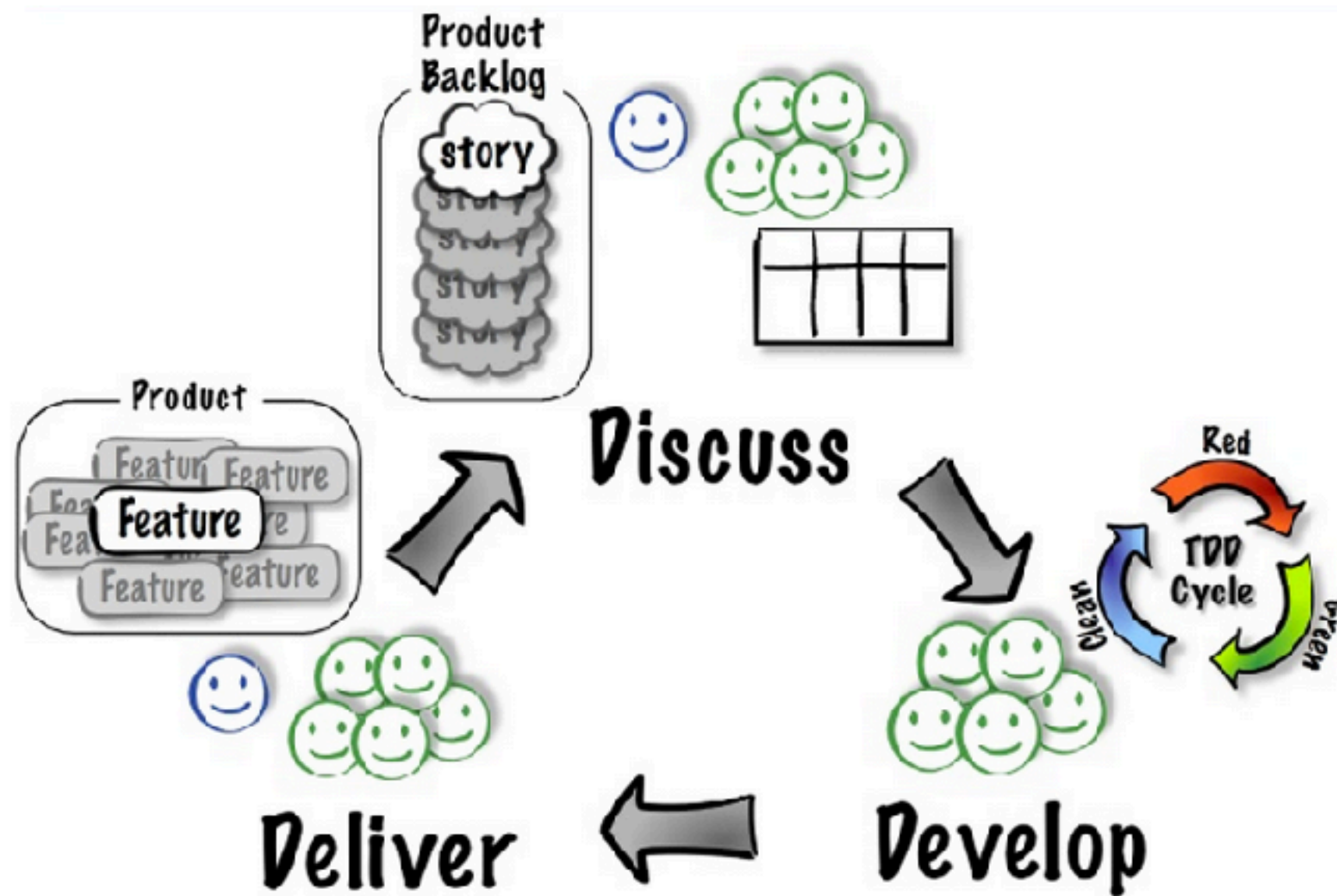
Design your pipeline/process



THINK before coding



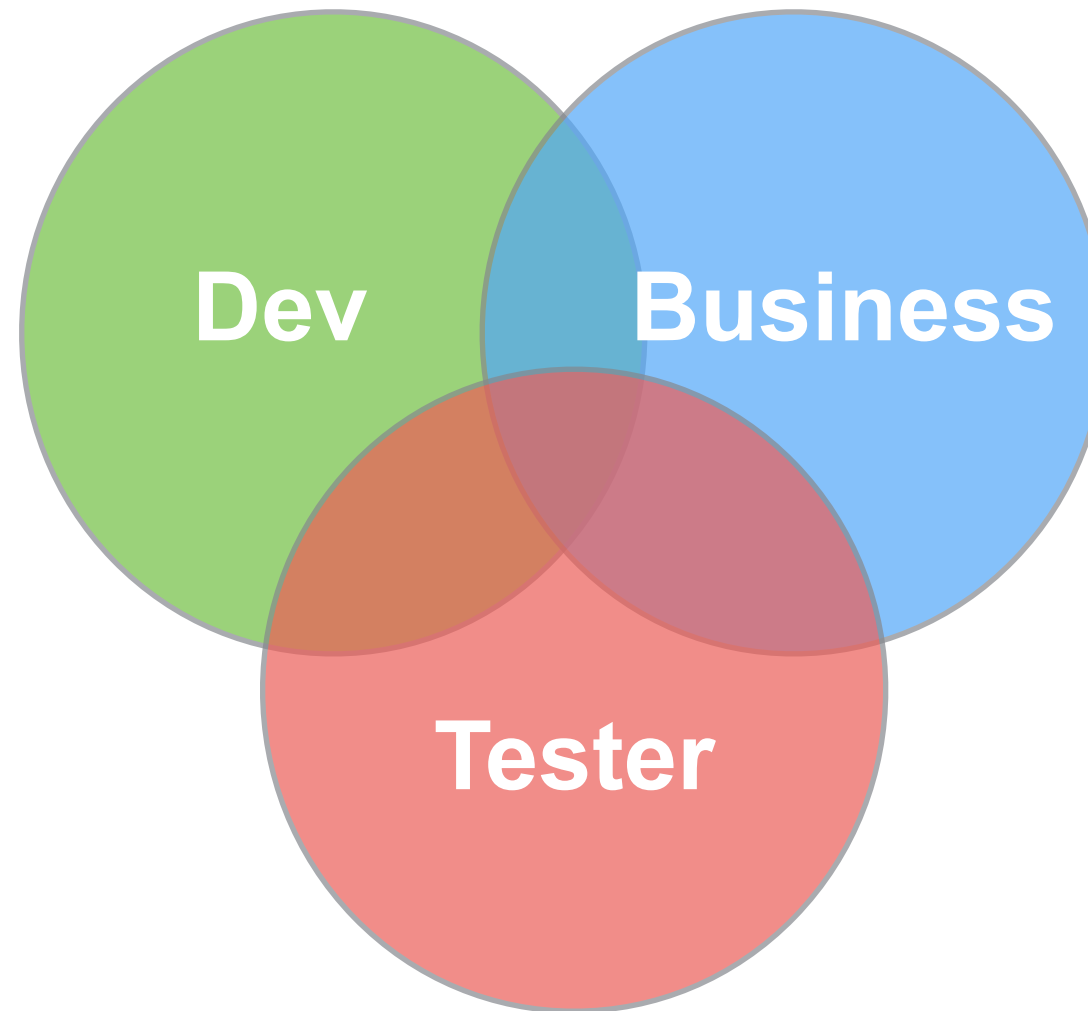
Acceptance Test-Driven Development



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



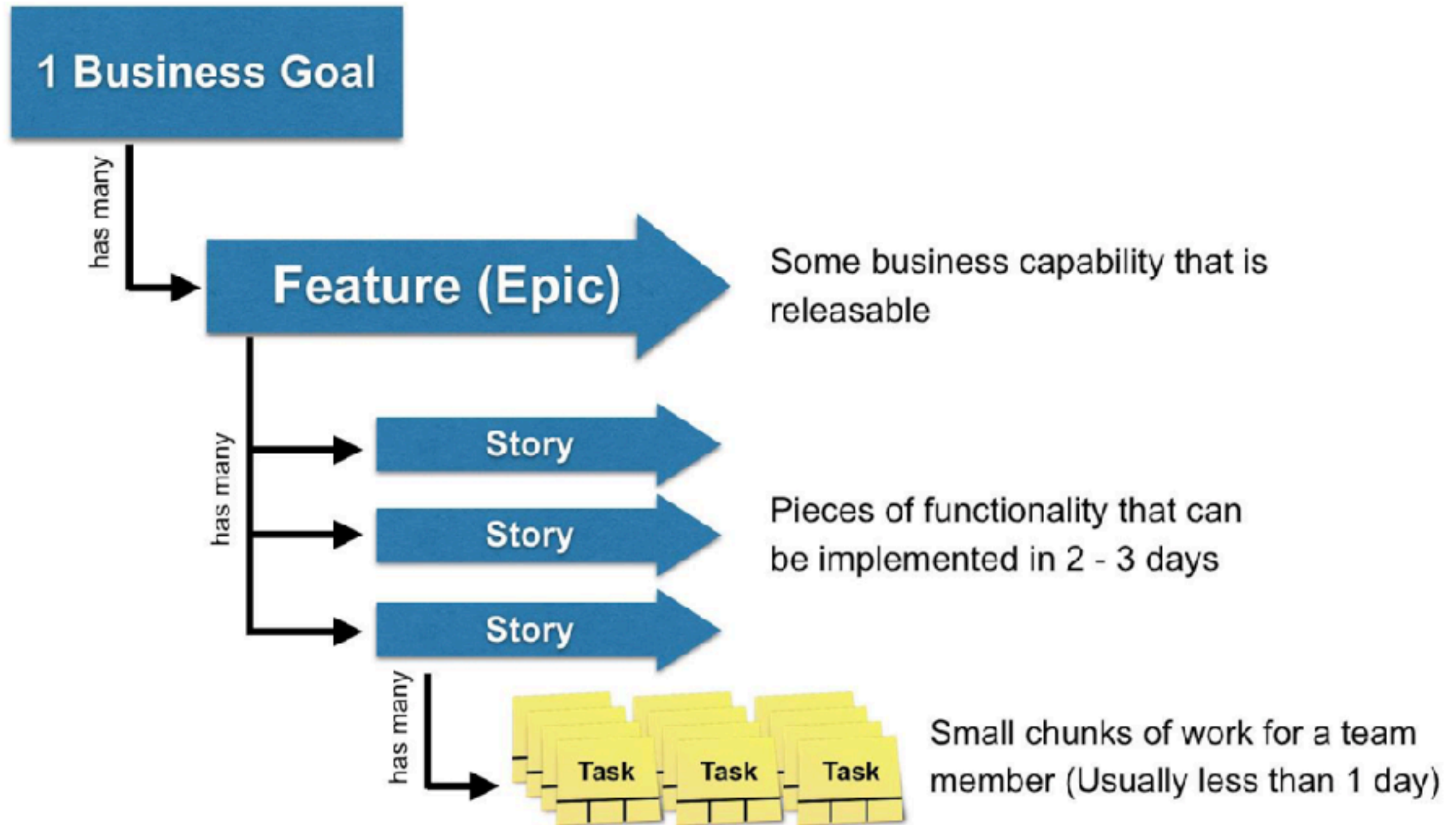
Acceptance Test-Driven Development



Acceptance Tests
=
Business Criteria
+
Examples (data)



Work break down



Iterative and incremental process

Feature 1

Time



Iterative and incremental process

Done = coded and tested



The diagram illustrates an iterative and incremental process. A horizontal black arrow points to the right, representing the progression of time. Above the arrow, on the left side, is a green rectangular box containing the text "Feature 1". This indicates that Feature 1 was completed early in the process. The word "Time" is written below the arrow, centered.

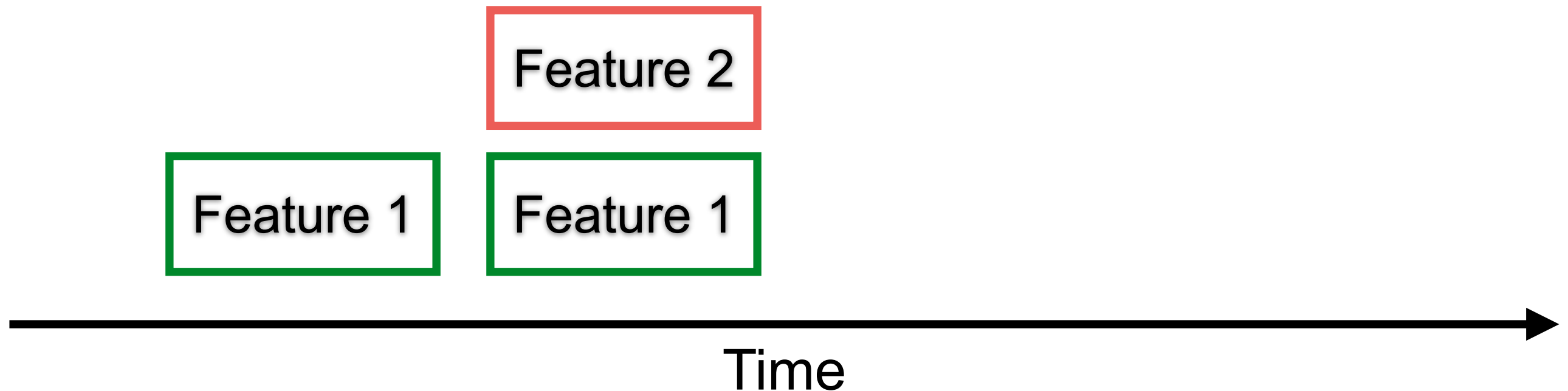
Feature 1

Time



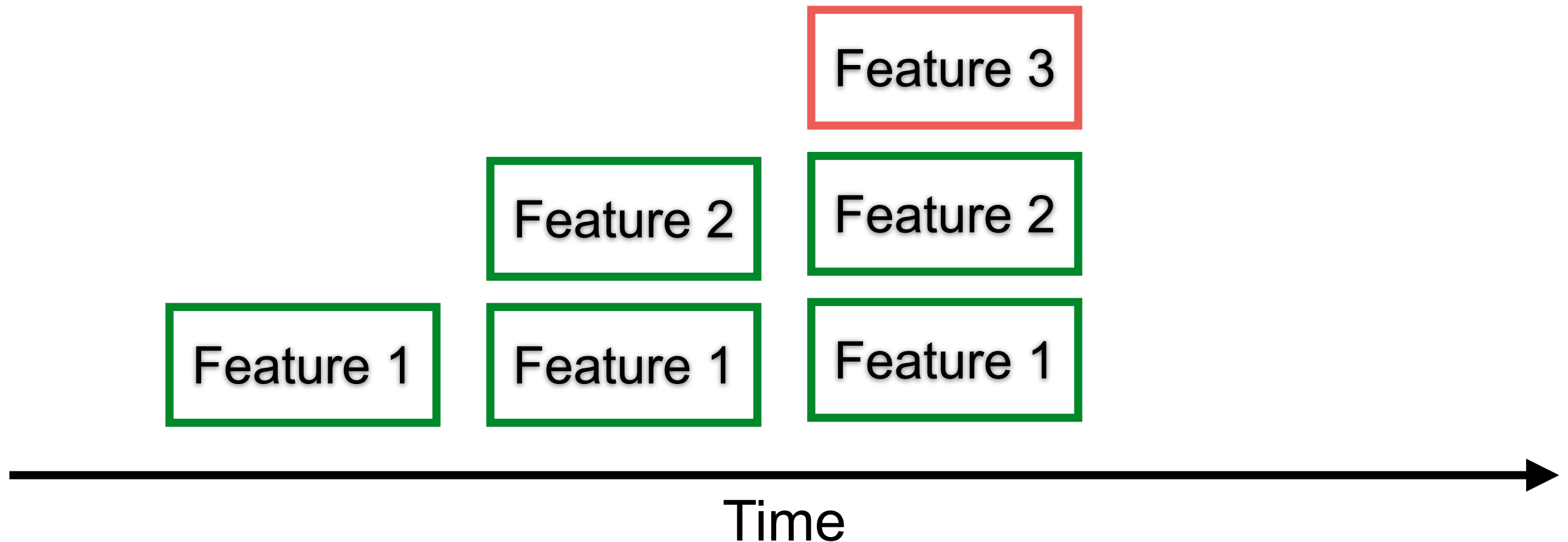
Iterative and incremental process

Done = coded and tested



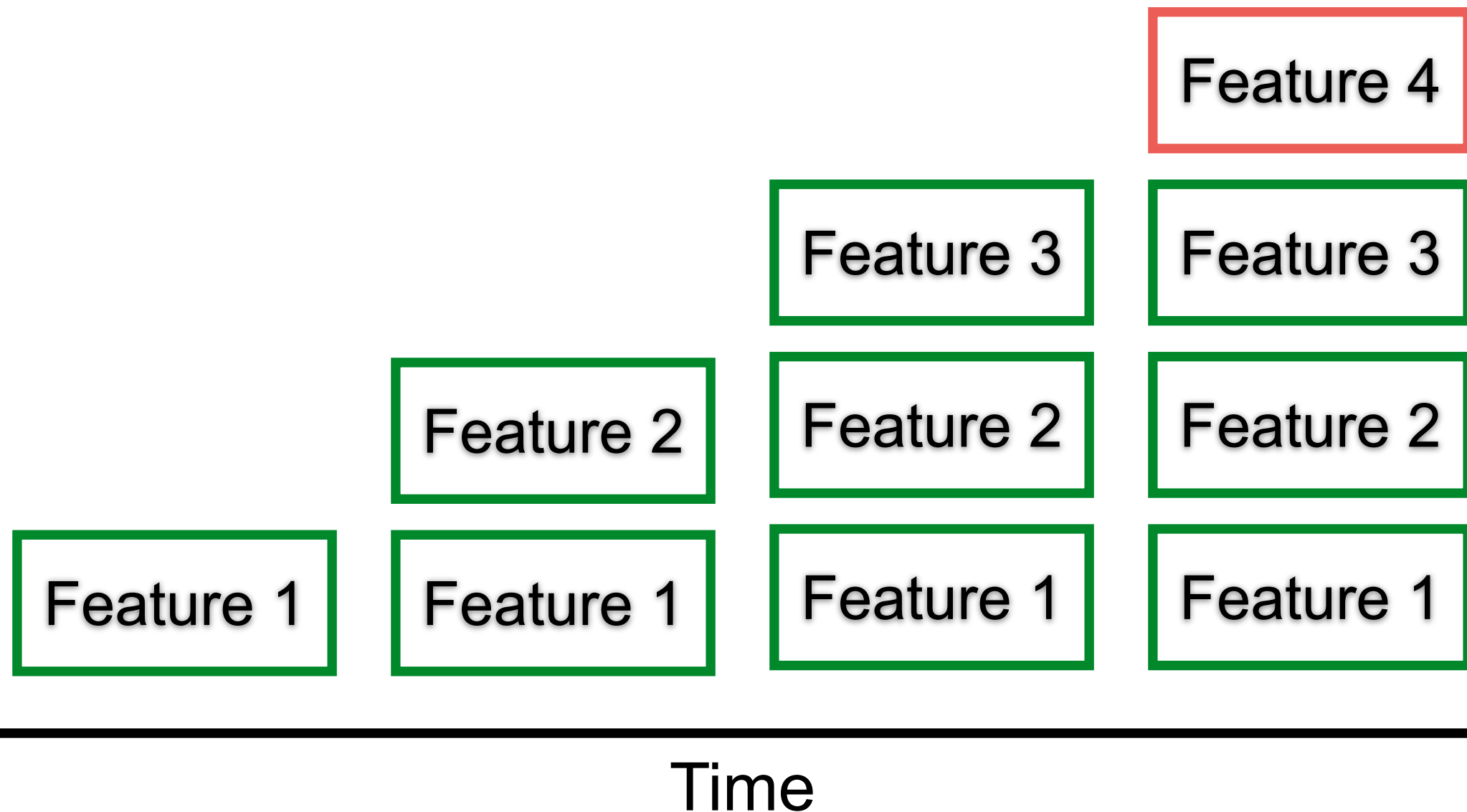
Iterative and incremental process

Done = coded and tested



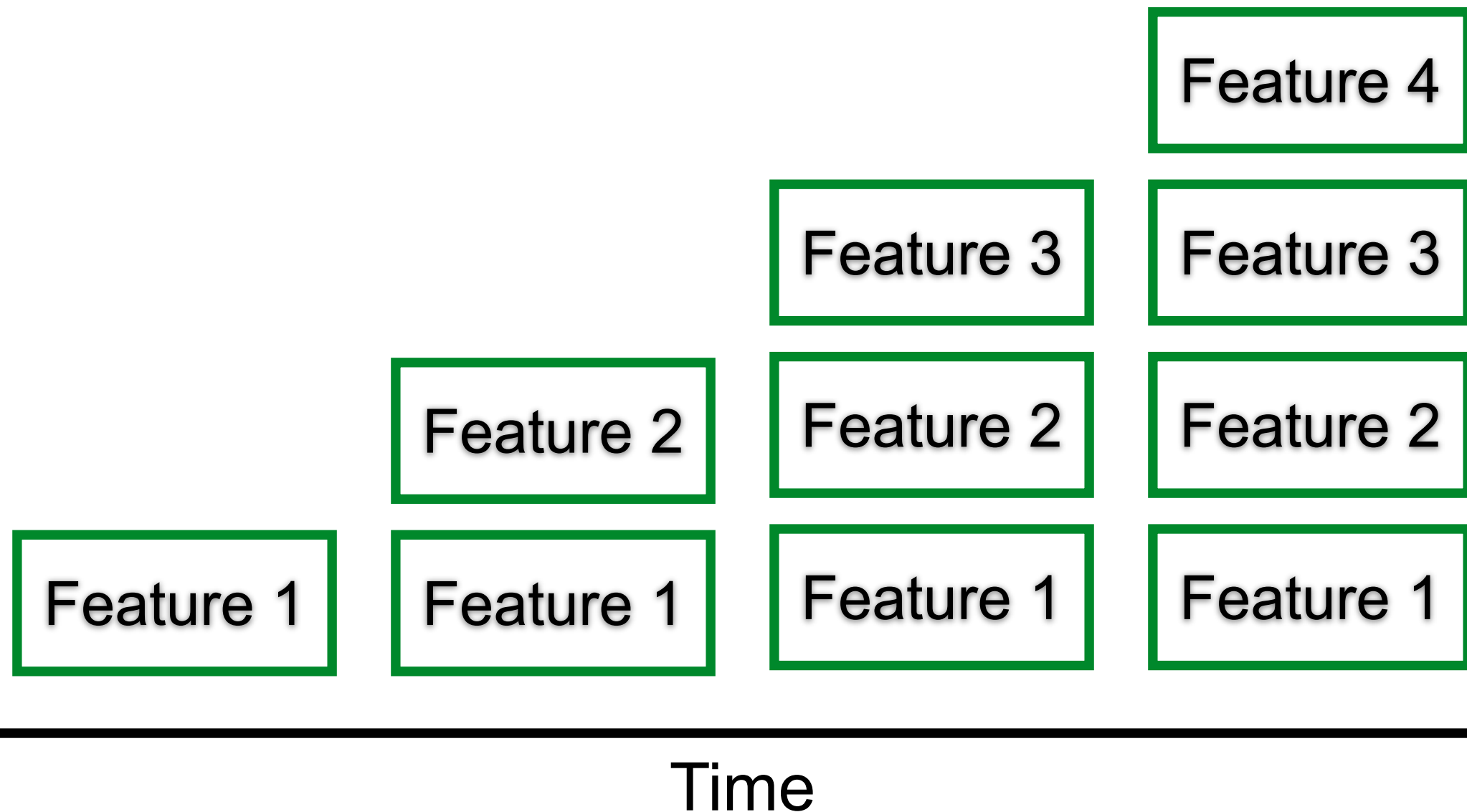
Iterative and incremental process

Done = coded and tested



Iterative and incremental process

Done = coded and tested



Iterative and incremental process

Done = coded and tested

