

Mastering Application Testing

Continuous UI Testing at Scale with Playwright



Let's Connect!



-  Randy-Pagels
-  PagelsR
-  <https://xebia.com>
-  @RandyPagels

Randy Pagels
DevOps Architect | Trainer
Randy.Pagels@Xebia.com



Thank you to our Sponsors!



Agenda

- Testing Challenges & Goals
- Playwright Overview
- Getting Started
- Authoring, Debugging, Running
- Additional Built in Features
- Time Travel Experience
- Continuous Integration Testing
- Playwright Testing Service



UI Testing Challenges

Consistent design across browsers!

Dynamic content and maintaining performance!

- Testing is hard
- Testing takes time to learn
- Testing takes time to build
- Tests are slow – they take too long to run!
- Tests are flaky – they crash all the time!
- Tests are brittle – they break whenever the app changes!



Modern Testing Goals

Major goals for modern web testing

- Make testing easy!
- Make testing fun!
- Focus on fast feedback loops!
- Make test development as painless as possible!
- Testing tool that complements dev workflows!
- Testing that supports Testing At Scale!



Test Automation Practices

Tests should be automated when they give positive ROI

Advantages

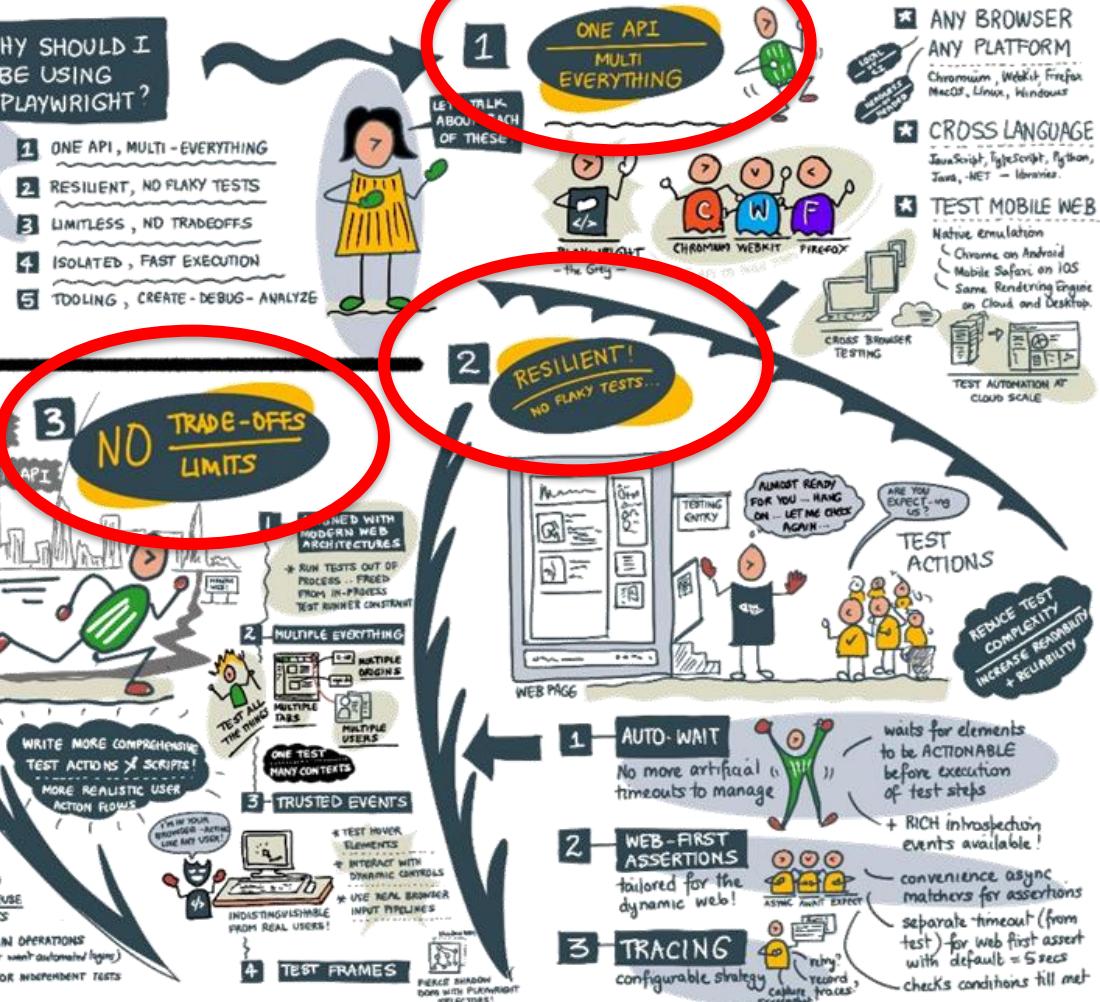
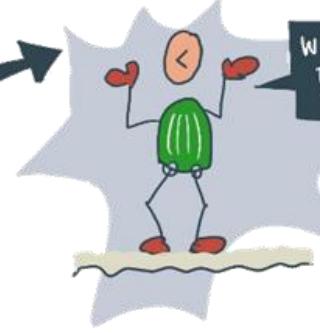
- Tests can be **rerun at any time** the same way.
- Tests can run as **part of CI/CD**.
- **Setup and cleanup** can be **automated**.
- Frameworks can provide reports, logs, and screenshots.

ProTips!

- Each **test** should **focus on one main behavior** or variation.
- **Tests** should **run independently** of each other and in any order.
- **Tests** should be **self-descriptive** and intuitively understandable.
- **Do not automate every test** – use a risk-based strategy with ROI.
- **Tests** should **run in parallel** at **scale**, especially in CI/CD.



AN INTRODUCTION TO PLAYWRIGHT



@playwrightweb
<https://playwright.dev>

ILLUSTRATED BY @SKETCHTHEDOCS

Playwright Overview

Open-source web test automation library on Node.js, which makes test automation easier for browsers based on Chromium, Firefox, and Webkit through a single API.

- ✓ **Cross-Browser**
 - Chromium, Firefox, and WebKit
- ✓ **Cross-language**
 - JavaScript, Python, and C#
- ✓ **Cross-platform**
 - Windows, Linux, and macOS
- ✓ **Modern Asynchronous API**
 - `async/await`
- ✓ **Rich Element Interaction**
 - wide range of built-in functions
- ✓ **Screenshots and Video Recording**
 - visual evidence of issues
- ✓ **Full Isolation – Fast Execution**
 - separate browser contexts



Reliable end-to-end testing for modern web apps

...by Microsoft

...and the people who made Puppeteer

<https://playwright.dev/>

<https://github.com/microsoft/playwright>

Playwright Architecture

✓ Language Bindings



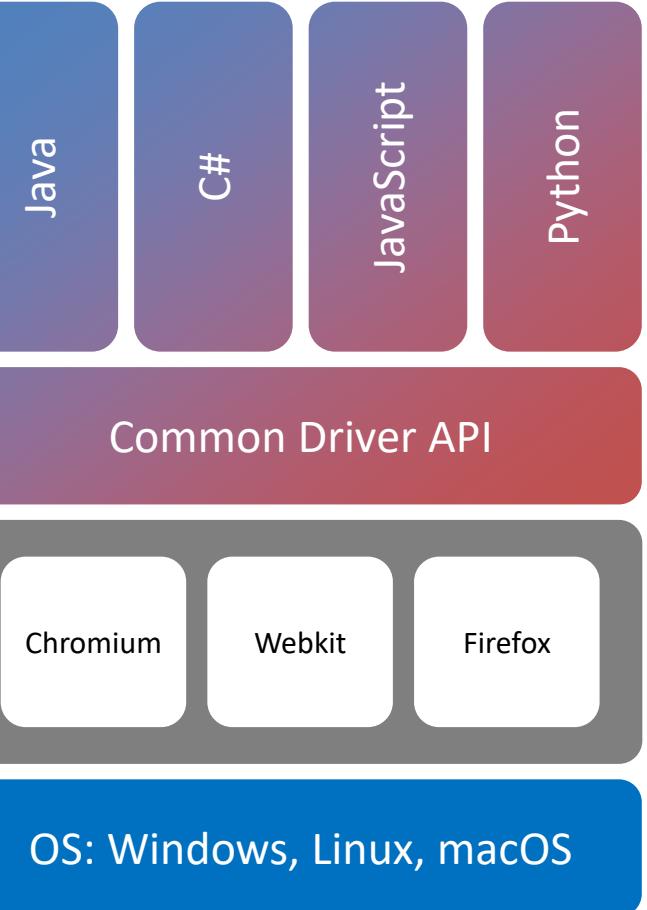
✓ Single Automation Protocol

✓ Unify all debugging protocols



✓ (Native) browser debugging e.g. ChromeDevTools Protocol, protocols,

(Safari) Web Inspector



Getting Started

Setting up using VS Code

Install Extension

Install Playwright

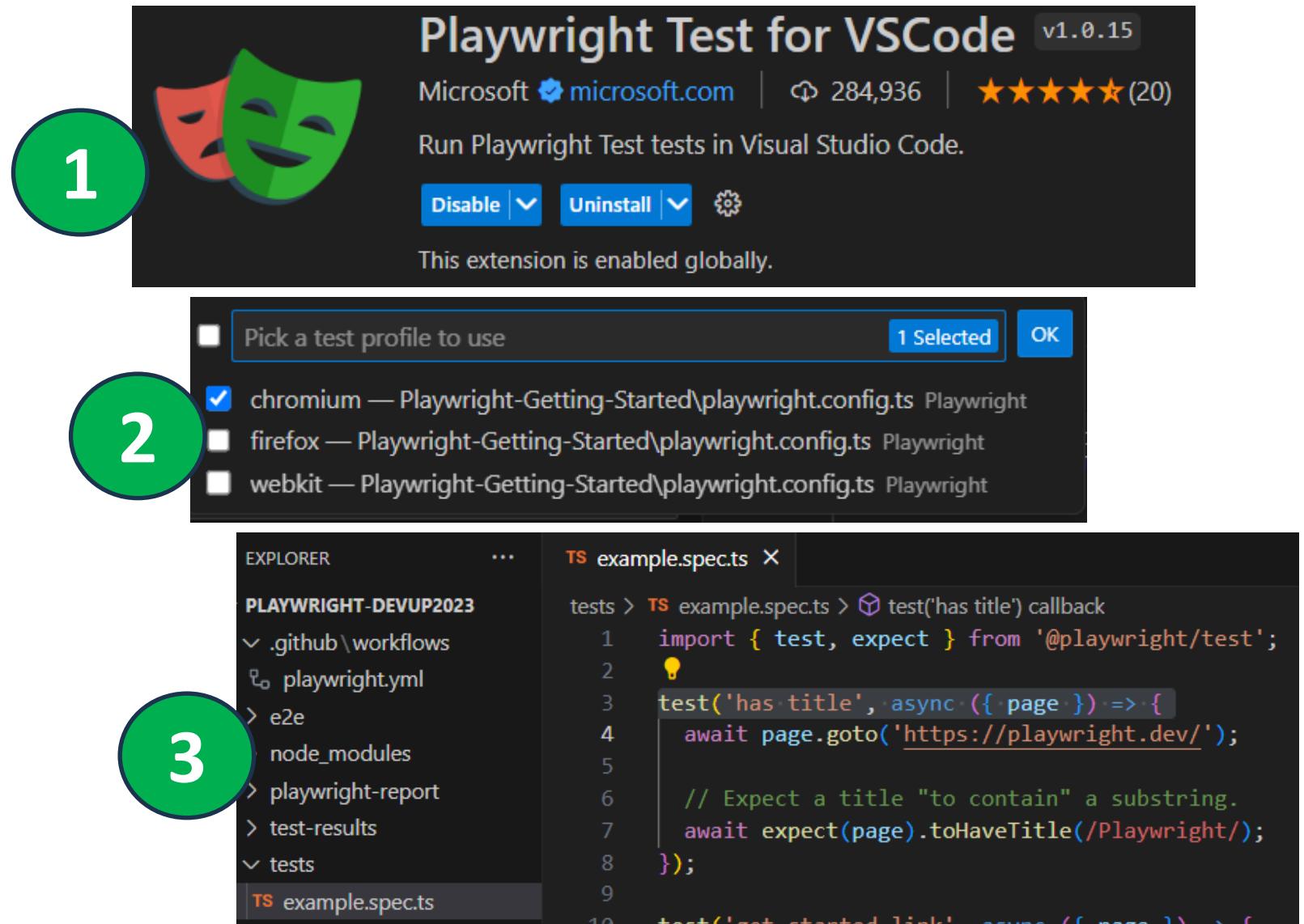
- npm init playwright@latest

Run Tests

- npx playwright test
- npx playwright test --ui

Test Reports

- npx playwright show-report



Test Generator

Playwright Test Generator is a GUI tool that records actions performed in the browser and generates code.

CLI

- npx playwright codegen

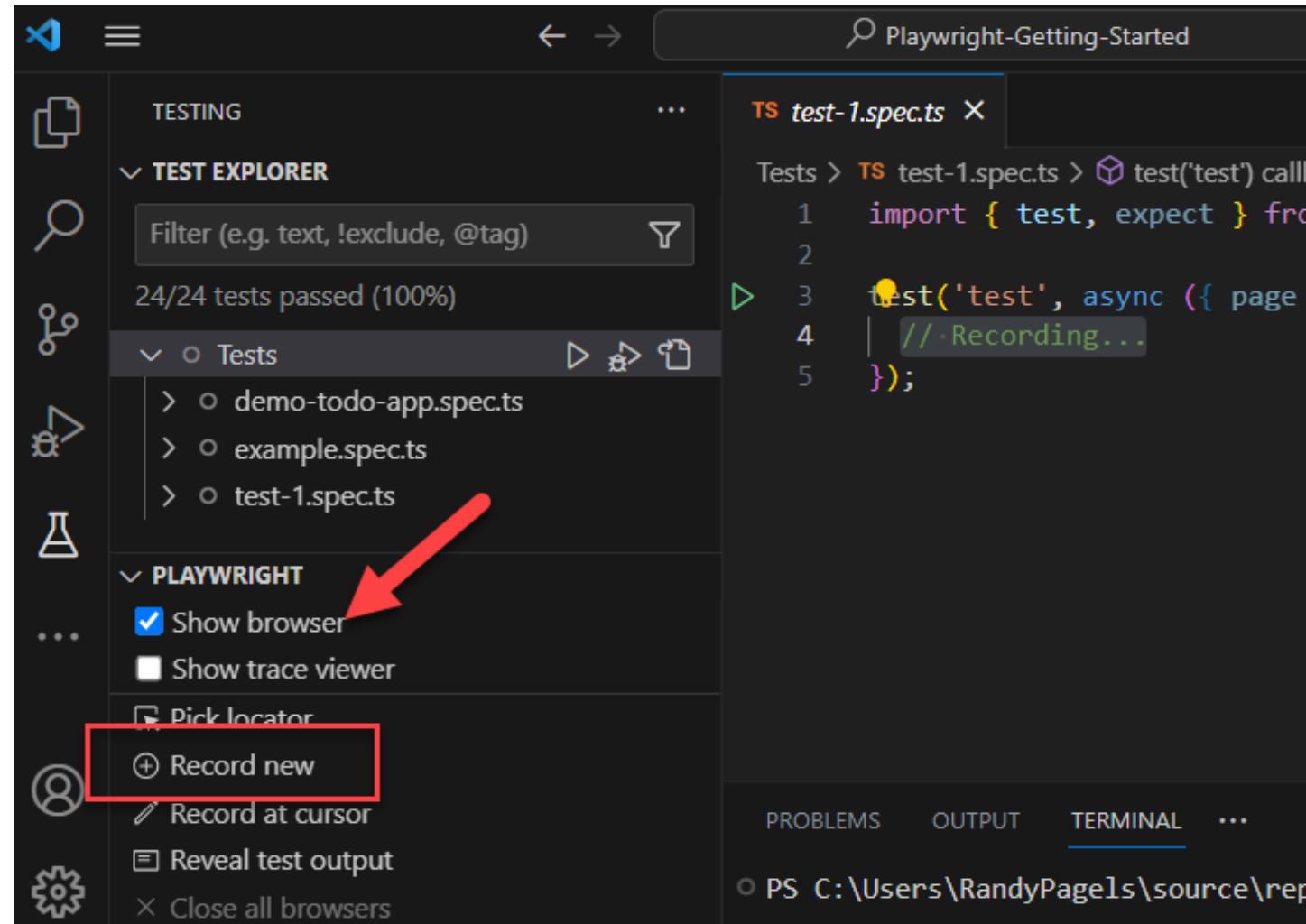
VS Code

- Use Plugin, "Record Now"

With the test generator you can:

- Record Actions
- Assert Elements
- Create Locators
- Emulate Devices

<https://playwright.dev/docs/codegen>



Trace Viewer

Playwright Trace Viewer is a GUI tool that helps you explore recorded Playwright traces after the script has ran.

CLI

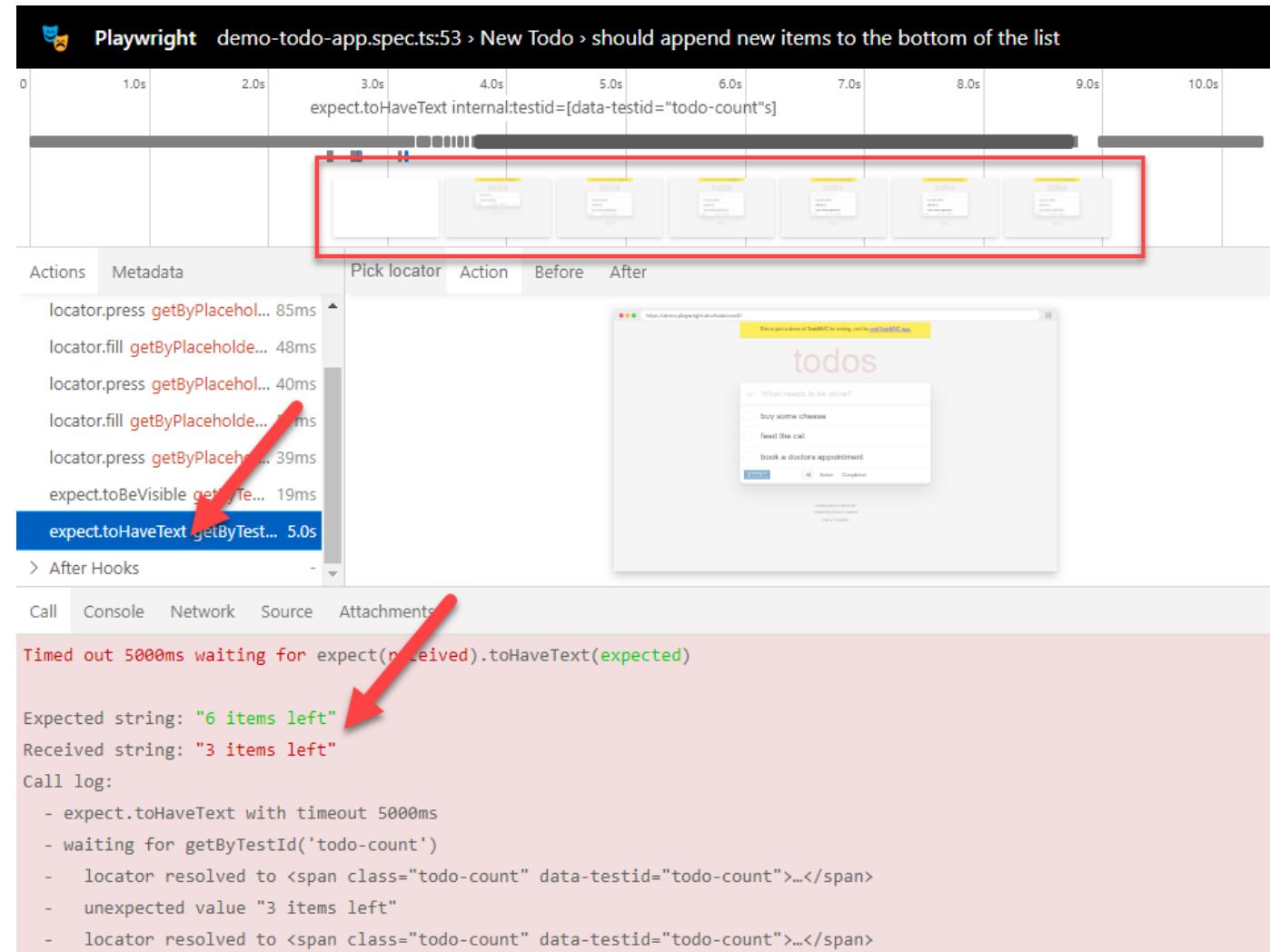
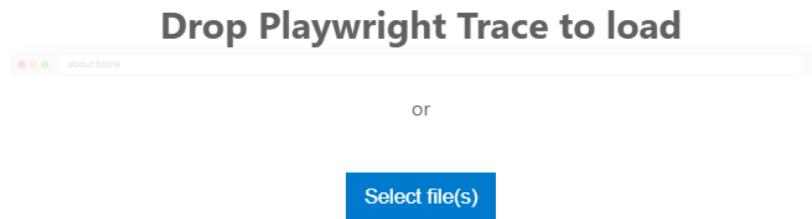
- `npx playwright test test.ts --trace on`

VS Code

- Use Plugin

Hosted Trace File Viewer

- <https://trace.playwright.dev>



Report List

Playwright Test comes with a few built-in reporter.

HTML Report

- npx playwright test --reporter=html

List Report

- npx playwright test --reporter=list

Line Report

- npx playwright test --reporter=line

Dot Report

- npx playwright test --reporter=dot

1

All 24 Passed 23 Failed 1 Flaky 0 Skipped 0

Project: chromium Total time: 12.0s

demo-todo-app.spec.ts

- ✗ New Todo › should append new items to the bottom of the list 7.2s
demo-todo-app.spec.ts:53
- ✓ New Todo › should allow me to add todo items 1.3s
demo-todo-app.spec.ts:14
- ✓ New Todo › should clear text input field when an item is added 1.4s
demo-todo-app.spec.ts:40

2

```
Running 6 tests using 6 workers

✓ 1 [chromium] > demo-todo-app.spec.ts:14:7 > New Todo >
✓ 2 [chromium] > demo-todo-app.spec.ts:53:7 > New Todo >
✓ 3 [firefox] > demo-todo-app.spec.ts:14:7 > New Todo > 5
✓ 4 [firefox] > demo-todo-app.spec.ts:40:7 > New Todo > 5
✓ 5 [chromium] > demo-todo-app.spec.ts:40:7 > New Todo >
✓ 6 [firefox] > demo-todo-app.spec.ts:53:7 > New Todo > 5

6 passed (11.1s)
```

3

```
Running 6 tests using 6 workers
.....
6 passed (11.1s)
```

Playwright UI Mode

Explore, run and debug tests with a time travel experience complete with watch mode

CLI

```
npx playwright test testname.ts --ui
```

DOM Inspection

- page structure, inspect elements, view attributes.

Console Logs

- JavaScript-generated logs, errors, and warnings.

Network Monitoring

- Requests, timings, inspect headers, payloads.

Interactivity

- Trigger events, clicks, real-time page responses.

Element Highlighting

- Hover over elements to highlight them.

Source Code Access

- View HTML, CSS, JS source for debugging.

The screenshot shows the Playwright UI Mode interface. At the top, there's a timeline from 0 to 20.0s. Below it, a tree view shows a test file 'demo-todo-app.spec.ts' with a 'New Todo' section containing three passed tests: 'should allow me to add todo items', 'should clear text in input field', and 'should append new item to list'. A red box highlights this section. Below the tree view is a list of test cases: 'Item', 'Editing', 'Counter', 'Clear completed button', 'Persistence', and 'Routing'. At the bottom, a code editor shows the source code for the 'New Todo' test, with a red arrow pointing to the 'test.describe' line. The code is as follows:

```
1  import { Page } from 'playwright';
2  import { test, expect } from '@playwright/test';
3
4  test('should allow me to add todo items', async ({ page }) => {
5    // Create a new todo input
6    const input = page.locator('#new-todo');
7    await input.fill('Buy some cheese');
8    await input.press('Enter');
9    expect(page.textContent).toContain('feed the cat');
10   expect(page.textContent).toContain('book a doctors appointment');
11 });
12
13 test.describe('New Todo', () => {
14   test('should allow me to add todo items', async ({ page }) => {
15     // Create a new todo input
16     const input = page.locator('#new-todo');
```

Demo

- ✓ Getting Started
- ✓ Generate Tests
- ✓ Debugging
- ✓ Trace Viewer
- ✓ Test Report
- ✓ UI Mode – Visual Feedback



Automating Manual Testing



Azure DevOps Test Plans with Playwright

Publish your automated Playwright test results to Azure Test Plans

- ✓ Associate automated tests with Test Cases!
- ✓ Using Azure Test Plans with Playwright!

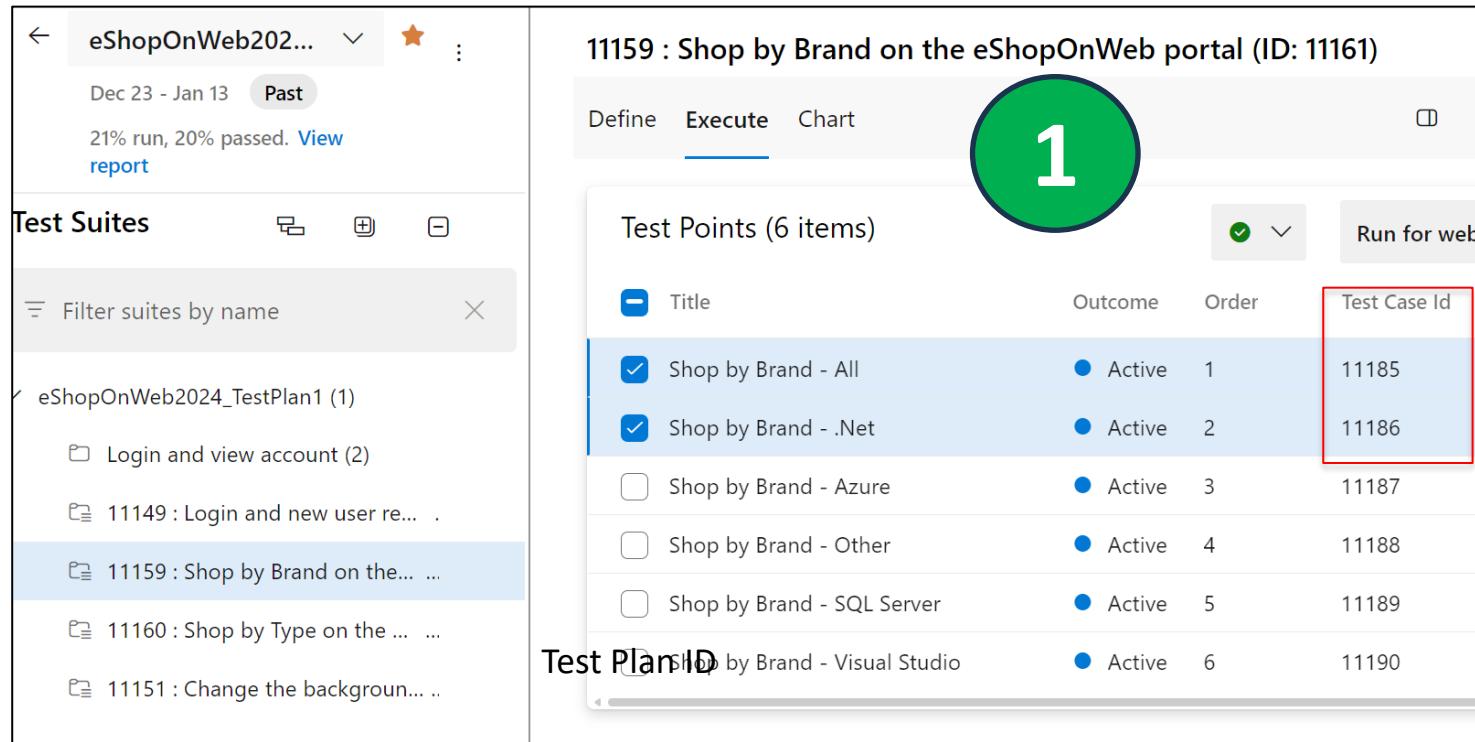
- **End-to-End Traceability:** Requirements (PBI's, User Stories) in Azure Boards are linked to test cases.
- **User-Friendly Testing:** Pipelines enabling the triggering of automated tests through Test Cases, rather than pipelines.
- **History Access:** Azure Test Plans offers access to historical test data, simplifying progress tracking through reports and charts.
- **Test Inventory Management:** Both automated and manual tests are managed efficiently, enabling effective tracking of their status and the progress of automation efforts.

Test Case Setup

Publish your automated Playwright tests to the ADO service

Manually create new test cases in Azure Test Plans

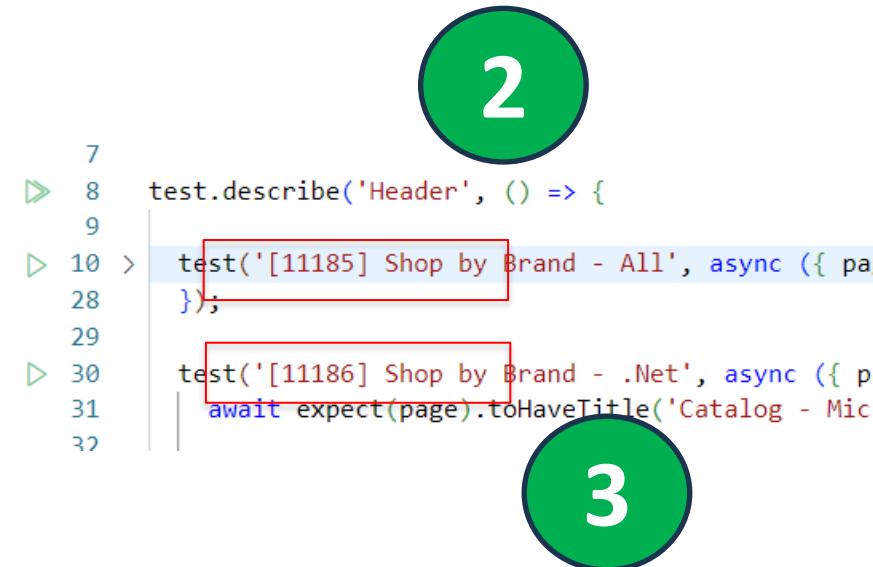
1. Create manual Test Cases
2. Add the **Test Case ID** in brackets to the test case title
3. Publish test results from build pipeline



A screenshot of the Azure Test Plans interface. On the left, there's a navigation pane with a dropdown for 'eShopOnWeb202...', a star icon, and a date range from 'Dec 23 - Jan 13' with a 'Past' button. Below that is a 'Test Suites' section with a 'Filter suites by name' input field. A list of suites includes 'eShopOnWeb2024_TestPlan1 (1)', '11149 : Login and new user re...', '11159 : Shop by Brand on the e...', '11160 : Shop by Type on the ...', and '11151 : Change the background...'. The '11159 : Shop by Brand on the e...' suite is selected and expanded. The main area shows the details for this suite, including its ID (11161) and title ('11159 : Shop by Brand on the eShopOnWeb portal (ID: 11161)'). Below this are tabs for 'Define', 'Execute' (which is selected), and 'Chart'. Under 'Execute', there's a 'Test Points (6 items)' section. A green circle with the number '1' is overlaid on this section. A table lists six test points with columns for 'Title', 'Outcome', 'Order', and 'Test Case Id'. The 'Test Case Id' column for the first two rows is highlighted with a red border: 'Shop by Brand - All' (id 11185) and 'Shop by Brand - .Net' (id 11186). The other four test points have empty checkboxes in the 'Title' column.

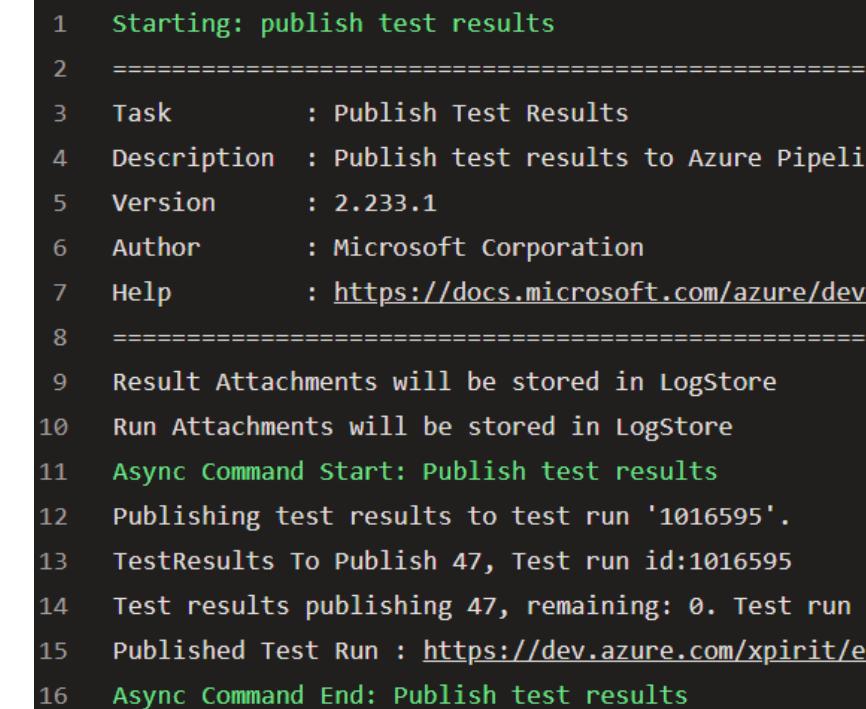
Title	Outcome	Order	Test Case Id
<input checked="" type="checkbox"/> Shop by Brand - All	● Active	1	11185
<input checked="" type="checkbox"/> Shop by Brand - .Net	● Active	2	11186
<input type="checkbox"/> Shop by Brand - Azure	● Active	3	11187
<input type="checkbox"/> Shop by Brand - Other	● Active	4	11188
<input type="checkbox"/> Shop by Brand - SQL Server	● Active	5	11189
<input type="checkbox"/> Test Plan ID by Brand - Visual Studio	● Active	6	11190

<https://marcusfelling.com/>



A screenshot of a Playwright test script. The script uses the Mocha framework with Chai assertions. It defines two test cases: 'Shop by Brand - All' (test ID 11185) and 'Shop by Brand - .Net' (test ID 11186). A green circle with the number '2' is overlaid on the first test case, and another green circle with the number '3' is overlaid on the second test case. The code is as follows:

```
7 > 8   test.describe('Header', () => {
9
10  > 10  > test('[11185] Shop by Brand - All', async ({ pa
28  });
29
30  > 30  > test('[11186] Shop by Brand - .Net', async ({ p
31  | await expect(page).toHaveTitle('Catalog - Mic
32
```



A screenshot of a build pipeline log. The log shows the 'Starting: publish test results' step, which is part of a task for publishing test results to Azure Pipelines. The task details are as follows:

Task	: Publish Test Results
Description	: Publish test results to Azure Pipelines
Version	: 2.233.1
Author	: Microsoft Corporation
Help	: https://docs.microsoft.com/azure/devops/

The log also indicates that result attachments will be stored in LogStore and run attachments will be stored in LogStore. It shows the start of publishing test results, the publishing process for test run 1016595, and the completion of publishing 47 test results. The final published test run URL is provided at the end.

```
1 Starting: publish test results
2 =====
3 Task      : Publish Test Results
4 Description: Publish test results to Azure Pipelines
5 Version   : 2.233.1
6 Author    : Microsoft Corporation
7 Help      : https://docs.microsoft.com/azure/devops/
8 =====
9 Result Attachments will be stored in LogStore
10 Run Attachments will be stored in LogStore
11 Async Command Start: Publish test results
12 Publishing test results to test run '1016595'.
13 TestResults To Publish 47, Test run id:1016595
14 Test results publishing 47, remaining: 0. Test run
15 Published Test Run : https://dev.azure.com/xpirit/e
16 Async Command End: Publish test results
```

Text Execution History

11149 : Login and new user regisraion to the eShopOnWeb portal using email id [?](#) and password (ID: 11156)

Define Execute Chart

Test Points (8 items)

Run for web app

1 Title

2 User Registration

3 User Account Login for General User

4 Invalid Login Attempt

5 User Account Login for Admin User

6

7

8

1 Failed

2 Passed

3 Failed

4 Failed

5 Passed

6

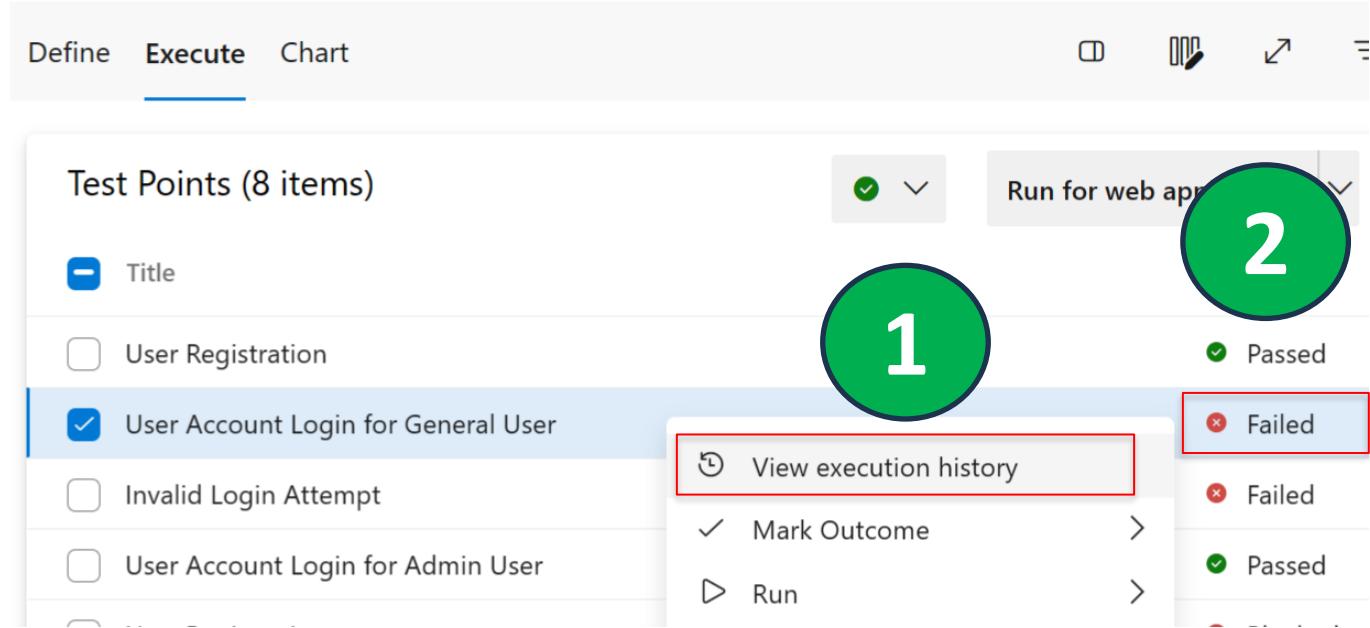
7

8

1 View execution history

2 Mark Outcome

3 Run



3

Test Case Results

Outcome	TimeStamp	Configuration	Run by
Failed	1h ago	Windows 10	Randy Pagels
Failed	1h ago	Windows 11	Randy Pagels
Passed	1h ago	Windows 11	Randy Pagels
Passed	1h ago	Windows 10	Randy Pagels
Passed	1h ago	Windows 10	Randy Pagels
Passed	1h ago	Windows 11	Randy Pagels
Passed	4h ago	Windows 11	Randy Pagels
Passed	4h ago	Windows 10	Randy Pagels
Passed	4h ago	Windows 10	Randy Pagels
Passed	4h ago	Windows 11	Randy Pagels
Passed	7h ago	Windows 11	Randy Pagels
Passed	7h ago	Windows 10	Randy Pagels
Passed	7h ago	Windows 11	Randy Pagels

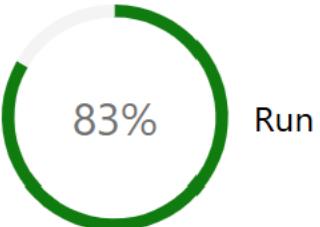
Test Plans Progress Report

Summary

 1 Test plans

 24 Test points

 20 (20 / 24) Test points run



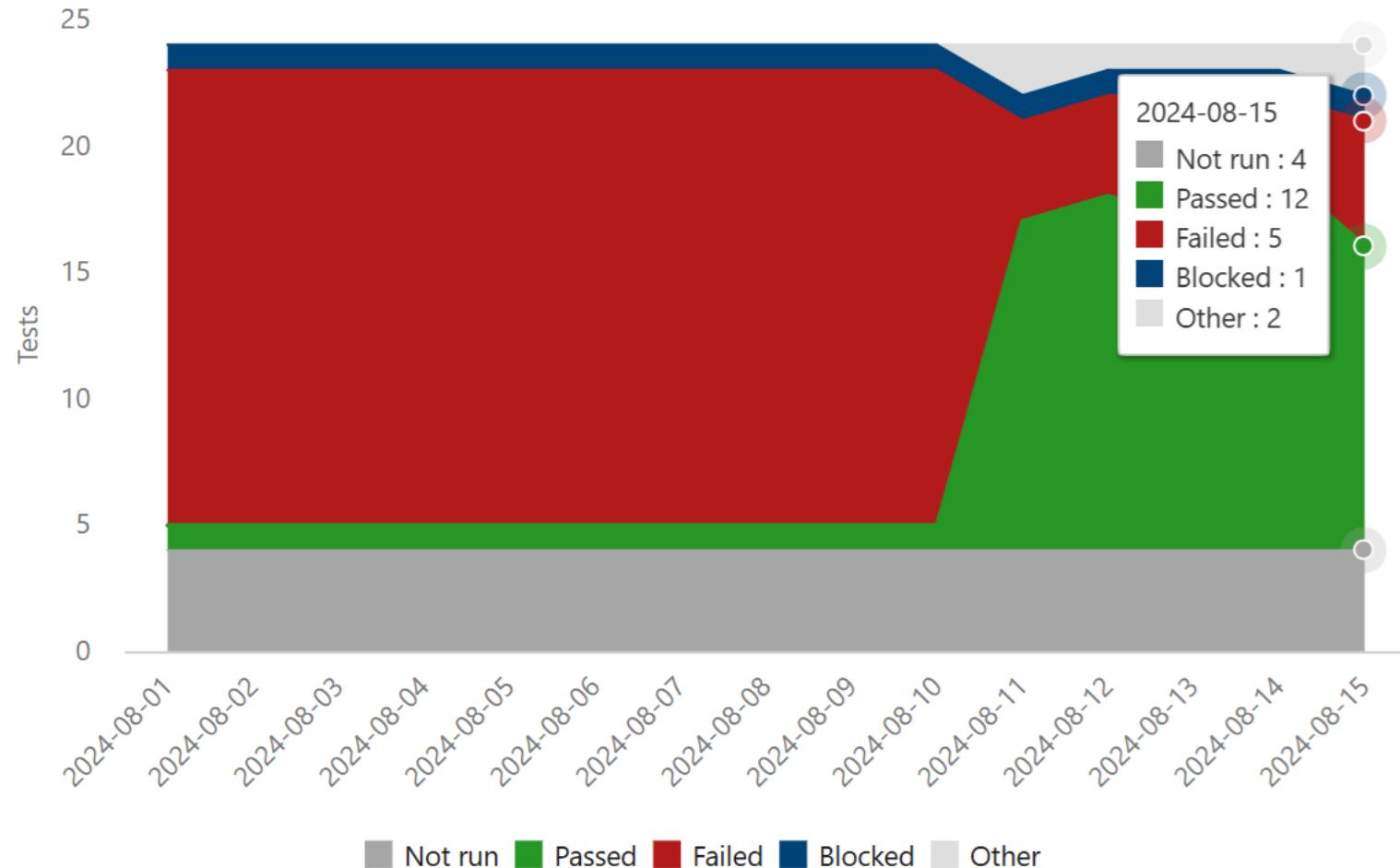
 60% (12 / 20) Pass rate



12  Passed

5  Failed

Outcome trend



Demo

- ✓ Automate Test Cases using Playwright
- ✓ Test Execution History
- ✓ Progress Report



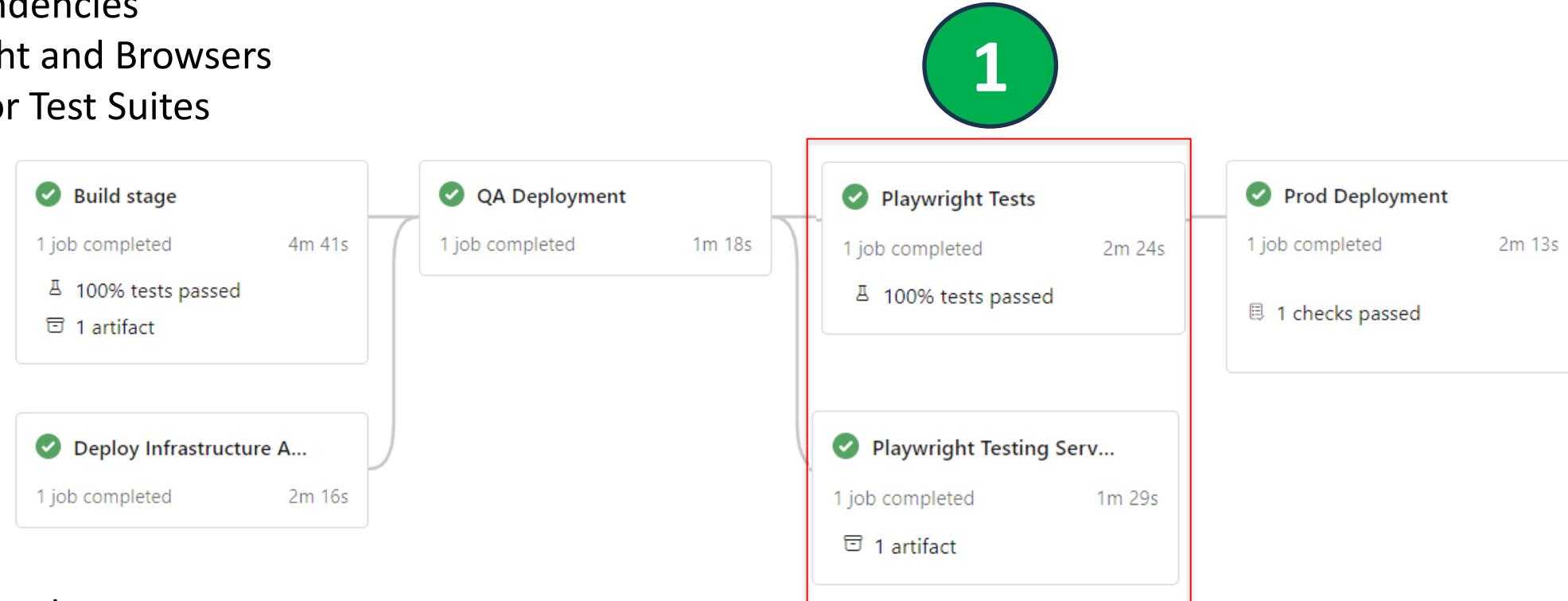
Continuous Integration Testing at Scale



Continuous Testing

Tests run on each push and pull request into the main/main branch

- ✓ Installs all dependencies
- ✓ Installs Playwright and Browsers
- ✓ Parallelization for Test Suites
- ✓ Run all the tests



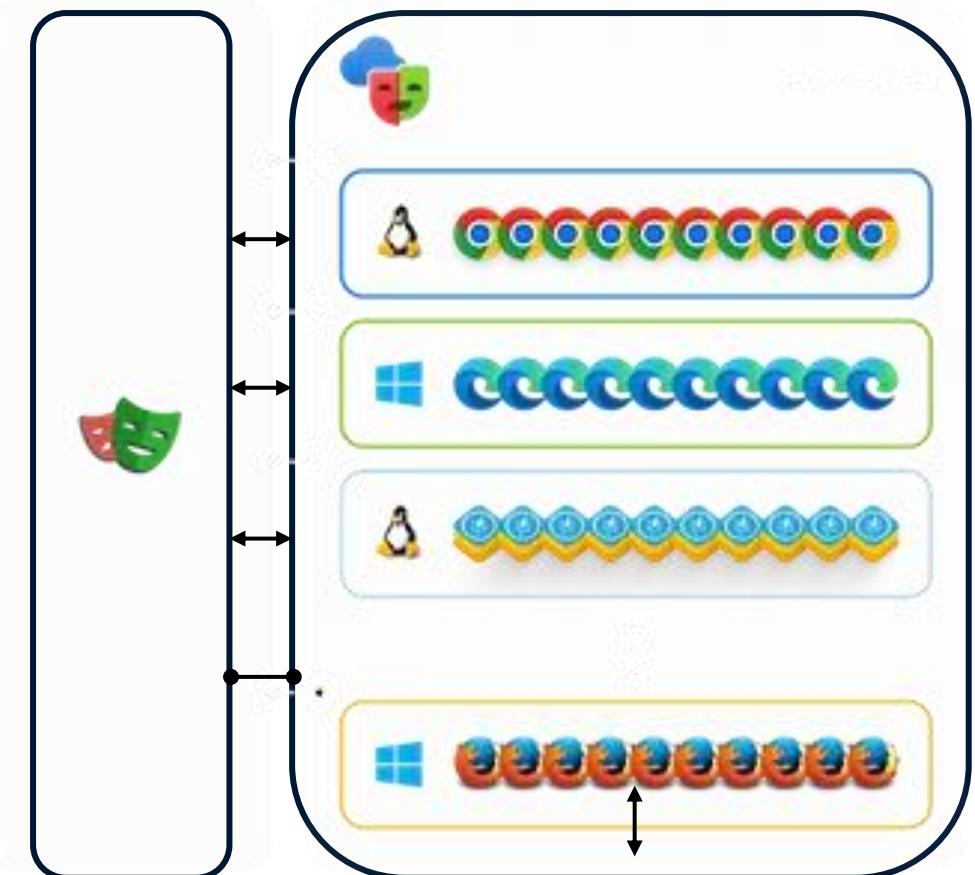
- ✓ Generate Test Results HTML report
- ✓ Generate Screenshots and Video Recordings
- ✓ Generate Debug Trace file
- ✓ Deploy static files to Website or GitHub Pages for viewing

2

Microsoft Playwright Testing

Scalable end-to-end modern web app testing service

- Run Playwright tests flexibly in the cloud
- Run Playwright tests with browser and OS combinations
- Cloud enabled to run Playwright tests at scale.
- High parallelization across operating system-browser combinations.
- Get tests done much faster.
- Speed up delivery of features without sacrificing quality.



Demo

- ✓ Build and Deploy Pipeline
- ✓ Pipeline Results
- ✓ Playwright Testing Service, Testing as Scale
- ✓ Playwright Testing Service Reporting (Private Preview)
- ✓ HTML Report on GitHub Pages for viewing



Pipeline Results

- Per build
1. Passed!
 2. Failed!
 3. Percentage!

Summary Tests Environments Advanced Security Alerts Scans Snyk Report Associated pipelines Code Coverage Mend WhiteSource Bolt Build Report

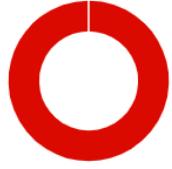
Summary

5 Run(s) Completed (4 Passed, 1 Failed) [4 unique failing tests in the last 14 days](#)

121
Total tests
+121



120 Passed
1 Failed
0 Others



1 Failed tests (+1)
1 New
0 Existing

99.17%
Pass percentage
↑ 99.1%

3m 51s
Run duration
↑ +3m 51s

0
Tests not reported

Bug ▾ Link Test run ▾ ⚙

Filter by test or run name Tags ▾ Test file ▾ Owner ▾

Test	Duration	Failing since	Failing build
✗ Playwright Tests from Azure Deploy Pipeline (47/47)	0:02:59.317		
✓ Header > [11190] Shop by Brand but Flaky	0:02:42.897		
✓ Header > [11179] Shop by Type but Flaky	0:01:23.136		
✗ Header > [11441] Login - Admin but flaky New	0:01:20.357	Monday	Current build
✓ Header > [11084] Login - General	0:00:16.354		

Microsoft Playwright Testing Rich Reporting

Test results filtered by failed and flaky tests

Reporting Private Preview

Update Azure Pipelines configuration for Playwright tests

(✖) Test run failed | (✓) 45 Passed | (✖) 3 Failed | (❓) 0 Flaky | (⊖) 0 Skipped

⌚ 1m:17.9s | 🏠 10th August 5:51 ...
⌚ 50 workers

Test summary

🔍 S:Failed ✖️ | S:Passed ✖️ | All (48) | Passed (45) | Failed (3) | Flaky (0) | Skipped (0) | Project

Test case	Project	Duration
ShopByBrand.spec.ts (7)		
✓ ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts > [11188] Shop by Brand - S...	chromium	⌚ 0m:22.9s
✓ ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts > [11186] Shop by Brand - S...	chromium	⌚ 0m:19.6s
✓ ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts > [11187] Shop by Brand - S...	chromium	⌚ 0m:15.6s
✓ ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts > [11189] Shop by Brand - S...	chromium	⌚ 0m:23.1s
✓ ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts > [11185] Shop by Brand - S...	chromium	⌚ 0m:17.9s
✖ ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts > [11190] Shop by Brand - S...	chromium	⌚ 1m:00s
✓ ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts > [11190] Shop by Brand - V...	chromium	⌚ 0m:18.8s

Microsoft Playwright Testing Rich Reporting

A test result with CI information

Reporting Private Preview

Microsoft Playwright Testing PREVIEW

Home / Test run / Test case

Update Azure Pipelines configuration

Test run failed | 45 Passed

Test summary

S:Failed X S:Passed X S:Flaky X

Test case

[11190] Shop by Brand but Flaky
Header>PlaywrightTests\ShopByBrand.spec.ts

TimedOut 1m:00s

Chrome windows chromium Update Azure Pi...

Errors Test steps Attachments

Test timeout of 6000ms exceeded.

Error: page.selectOption: Test timeout of 6000ms exceeded.
Call log:
- waiting for locator('#CatalogModel_BrandFilterApplied')
- locator resolved to <select class="esh-catalog-filter" id="CatalogModel_Bran...>...</select>
- selecting specified option(s)
- did not find some options - waiting...

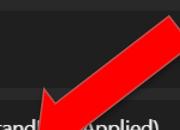
Test steps

Step	Duration
Before Hooks	16,731 ms
expect.toHaveTitle	144 ms
page.selectOption('#CatalogModel_BrandFilterApplied')	49,390 ms
After Hooks	10,993 ms

ShopByBrand.spec.ts (7)

ShopByBrand.spec.ts - Header>PlaywrightTests\ShopByBrand.spec.ts

eShopLogin.spec.ts (4)



Playwright Reporting Private Preview

Microsoft Playwright Testing is a service for running Playwright tests easily at scale. It uses the cloud to enable you to run Playwright tests with much higher parallelization across different operating system-browser combinations. The service is currently in preview.

Check out the Microsoft Playwright Testing service at <https://aka.ms/mpt/about>

Reporting Private Preview

Microsoft is building reporting feature for Microsoft Playwright Testing service to enable you to publish test results of your Playwright tests and view them in the service portal.

Join the waitlist for Microsoft Playwright Testing - [Reporting Private Preview](#)

Sign up at <https://aka.ms/mpt/reporting-signup>

Azure AD Authentication

- Store the credentials of test accounts using environment variables.

```
test(`example test`, async ({ page }) => {
  // ...
  await page.getLabel('User Name').fill(process.env.USERNAME);
  await page.getLabel('Password').fill(process.env.PASSWORD);
});
```

- Local Development? Use .env files and add to .gitignore.

```
# .env file
STAGING=0
USERNAME=me
PASSWORD=secret
```

- Set the values of these variables to use your CI system's secret management or Azure Key Vault.

```
testE2E:
  name: Run Playwright E2E tests
  runs-on: ubuntu-latest
  container: mcr.microsoft.com/playwright:v1.27.0-jammy
  env:
    USERNAME: ${{secrets.USERNAME}}
    PASSWORD: ${{secrets.PASSWORD}}
```

Note: MFA cannot be fully automated and requires manual intervention.

Tool Comparison

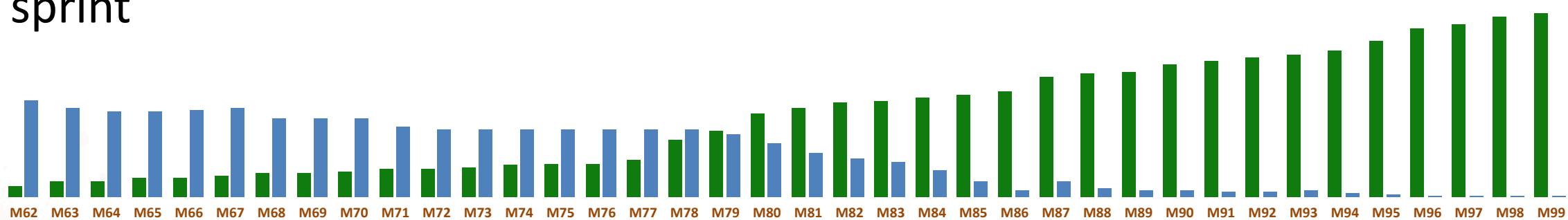
Playwright is not the only browser automation tool out there

Selenium	Cypress	Playwright
Popular among testers	Popular among developers	So hot right now
C#, Java, JavaScript, Ruby, Python	JavaScript only	C#, Java, JavaScript, Python
All major full browsers	Chrome, Edge, Firefox, Electron	Chromium+, Firefox, WebKit
WebDriver protocol	In-browser JavaScript	Debug protocols
Just a tool – requires a framework	Full, modern framework	Full, modern framework
Can be slow/flaky if waiting is poor	Visual execution but can be slow	Typically the fastest execution
Open source, standards, & gov	Open source from Cypress	Open source from Microsoft

Microsoft Azure DevOps team

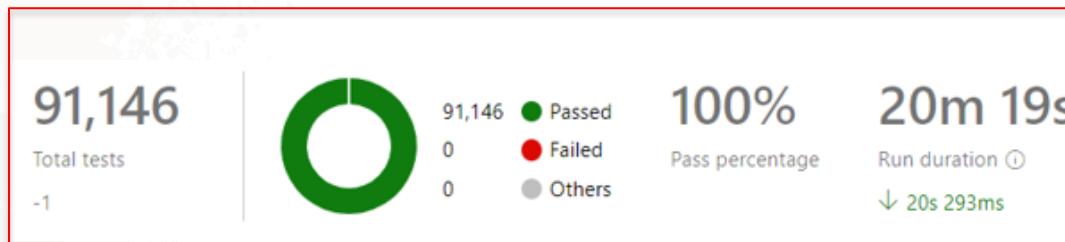
Balancing UI vs Unit Testing in your strategy to maximizes test coverage and effectiveness.

Long running UI functional tests at end of sprint



Shifted to unit tests from automated UI functional tests

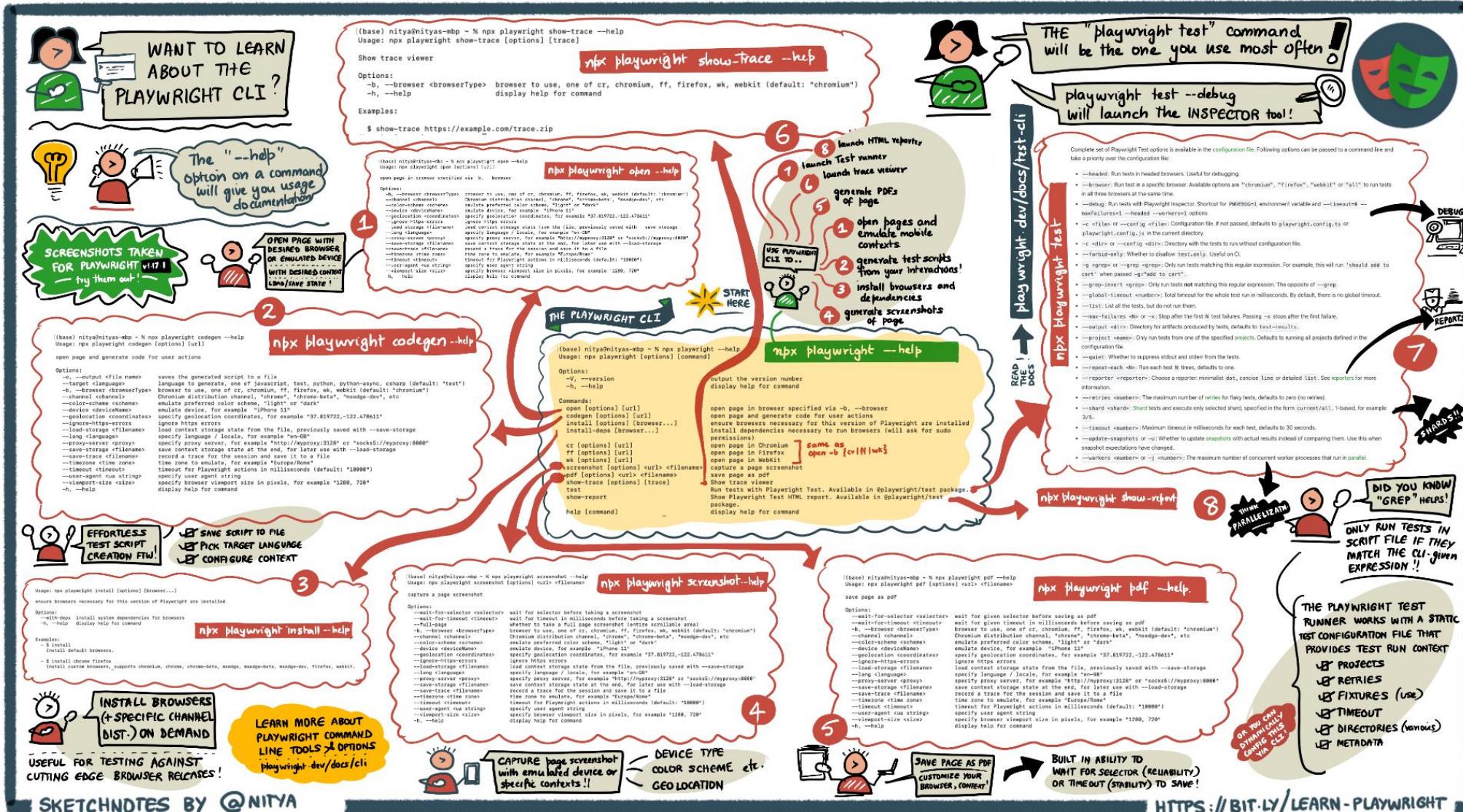
ADO PRs require 100% pass rate before merging



ADO Unit Tests in build pipeline for past 14 days



Playwright CLI



Playwright Resources

Documentation: <https://playwright.dev>

Source / Issues: <https://github.com/microsoft/playwright>

Social Media

- <https://aka.ms/playwright/discord>
- <https://aka.ms/playwright/youtube>
- <https://aka.ms/playwright/x>

Microsoft Playwright Testing

<https://aka.ms/mpt/about>

<https://playwright.microsoft.com/workspaces>

Test Automation with Playwright with Visual Guide Cheatsheet

<https://www.azurestaticwebapps.dev/blog/devtools-playwright>

eShopOnWeb: <https://github.com/dotnet-architecture/eShopOnWeb>

Slides: <https://github.com/PagelsR/Talks> (Mastering Application Testing with Playwright-v3.pdf)

Playwright Customer References

- **Mozilla:** Testing many projects, such as [Glean dictionary](#), [Firefox monitor](#), and [Firefox accounts](#).
- **Tesla:** starting using Playwright for e2e testing [September 2021](#).
- **WordPress:** migrated e2e and performance tests to Playwright March 2022. Blog post with details [here](#).
- **Adobe:** E2E for visual regression testing, and accessibility testing for [spectrum web components](#).
- **ING Bank:** uses Playwright to test their [web components](#), since [July 2020](#).
- **Facebook:** uses Playwright in their [React JS library](#).
- **NASA:** uses Playwright for end-to-end testing in [Open MCT \(Open Mission Control Technologies\)](#)
- **Johnson and Johnson:** [migrated to Playwright May 2022](#) to test their [Bodiless-JS Framework](#) for building editable websites on the JAMStack.
- **VMWare:** uses Playwright for integration testing [Tanzau Kubeapps](#).
- **Elastic:** [Test across their SaaS offerings](#) for search, logging, security, observability, and analytics.
- **Target:** uses Playwright to test [GoAlert](#) for on-call scheduling, automated escalations, and notifications.

Thank you!

Randy Pagels

DevOps Architect | Trainer
Randy.Pagels@Xebia.com

