Xebia | Microsoft Services

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

Xebia | Microsoft Services

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

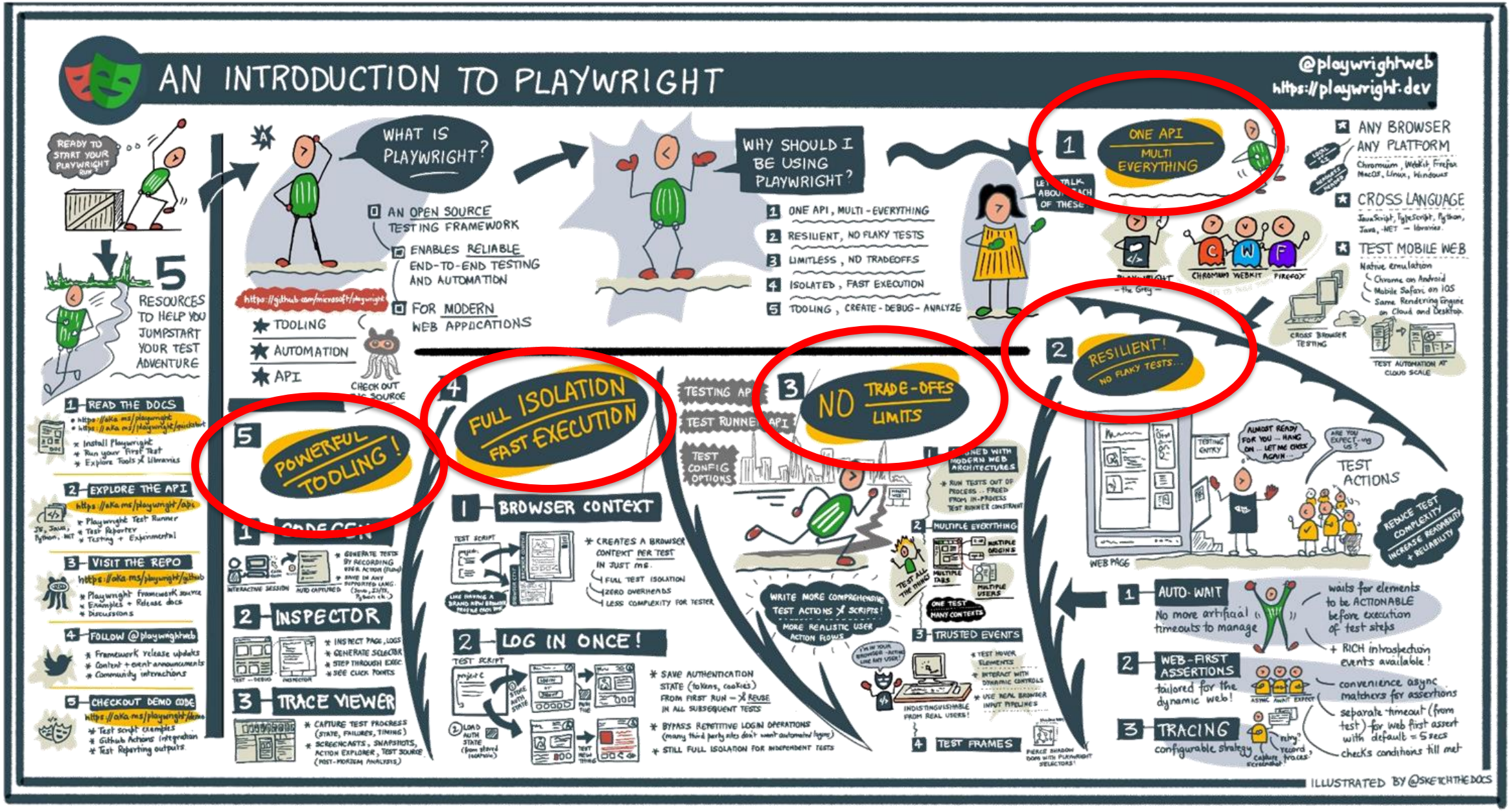
Advantages

- 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.

https://automationpanda.com



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

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/

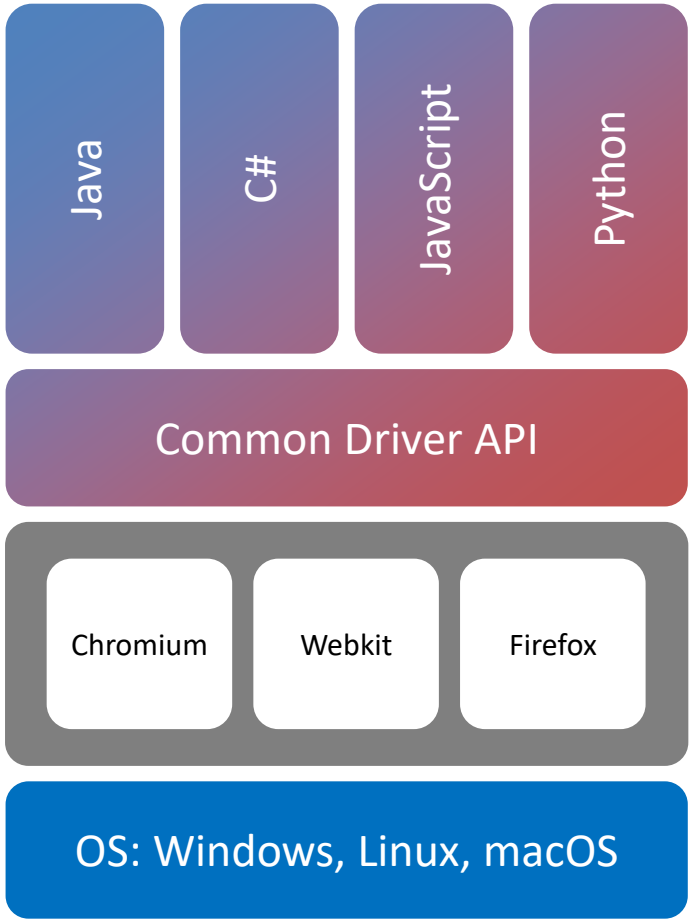
https://github.com/microsoft/playwright

Playwright Architecture

- Language Bindings →

- Single Automation Protocol
- Unify all debugging protocols →

- (Native) browser debugging protocols, e.g. ChromeDevTools Protocol, (Safari) Web Inspector →



Getting Started

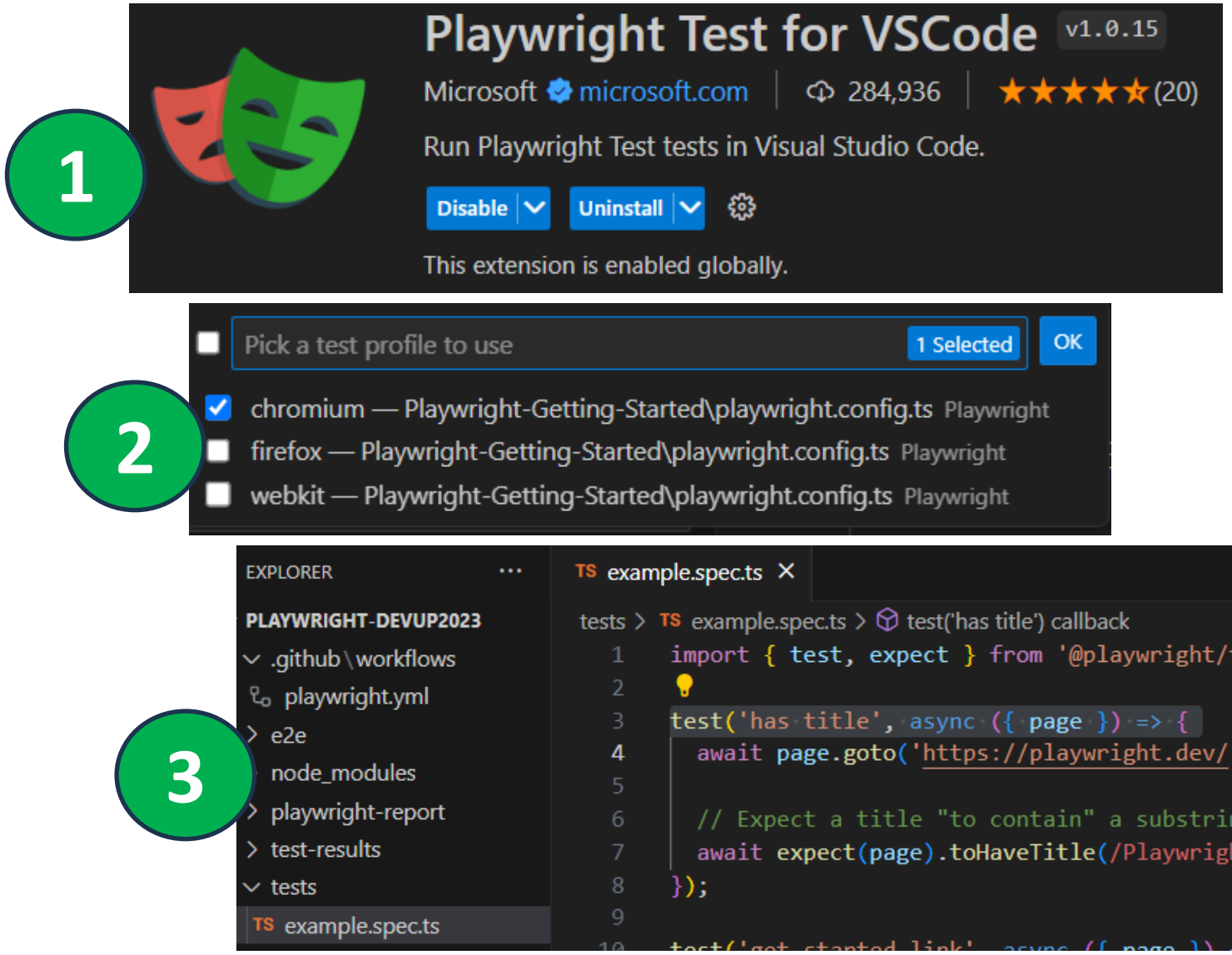
Setting up using VS Code

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 that records actions performed in the browser and generates code.

CLI

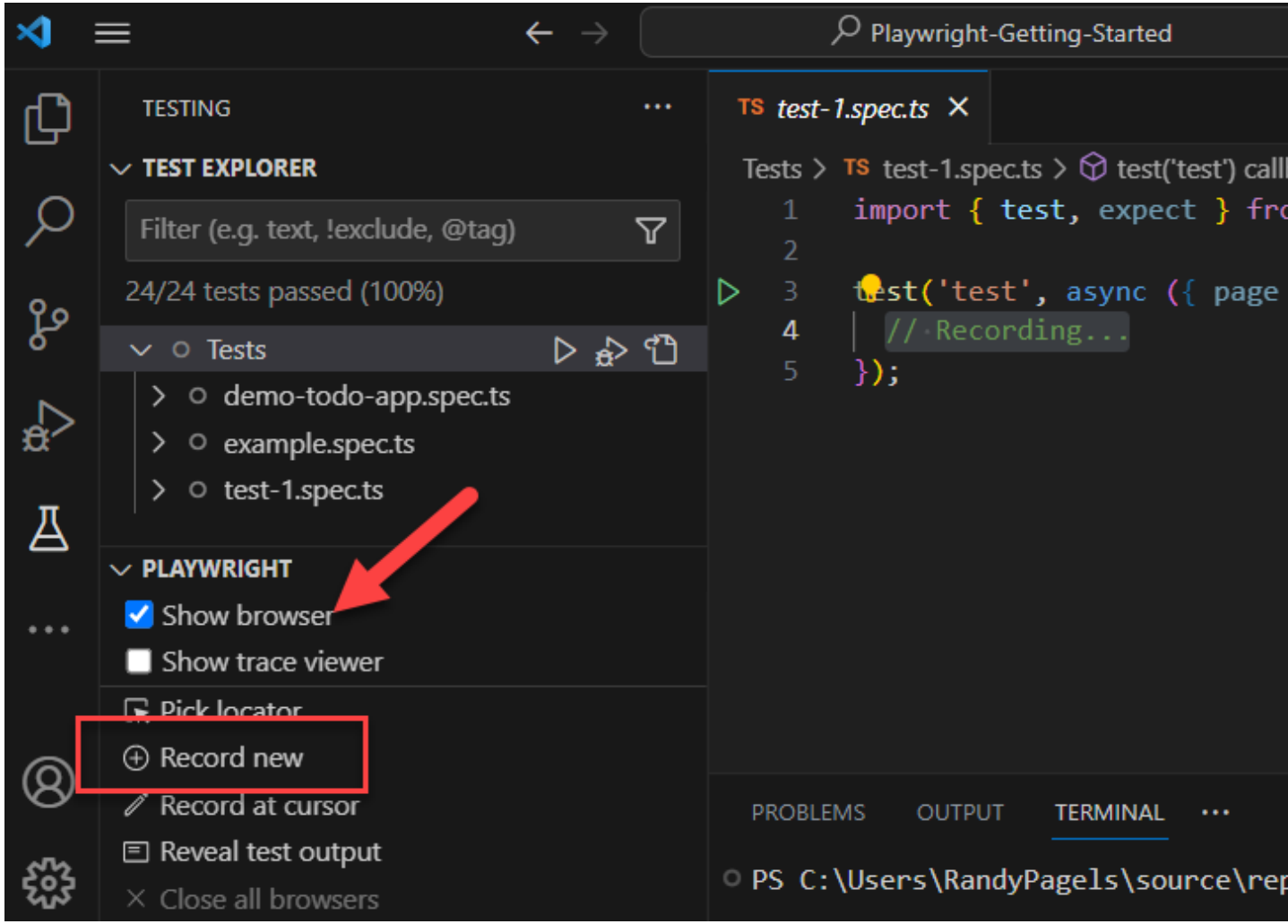
- npx playwright codegen

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

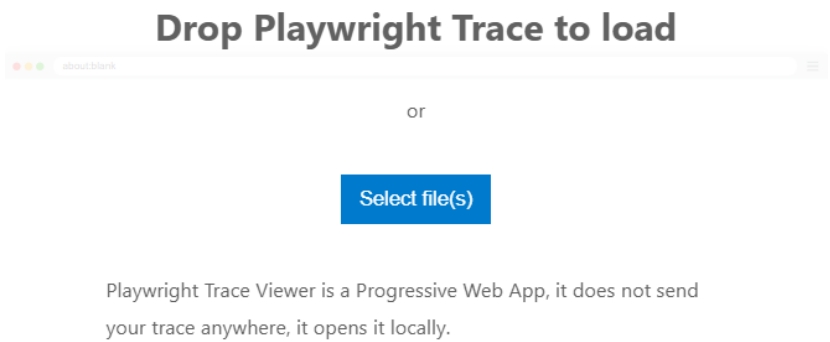
- npx playwright test test.ts --trace on

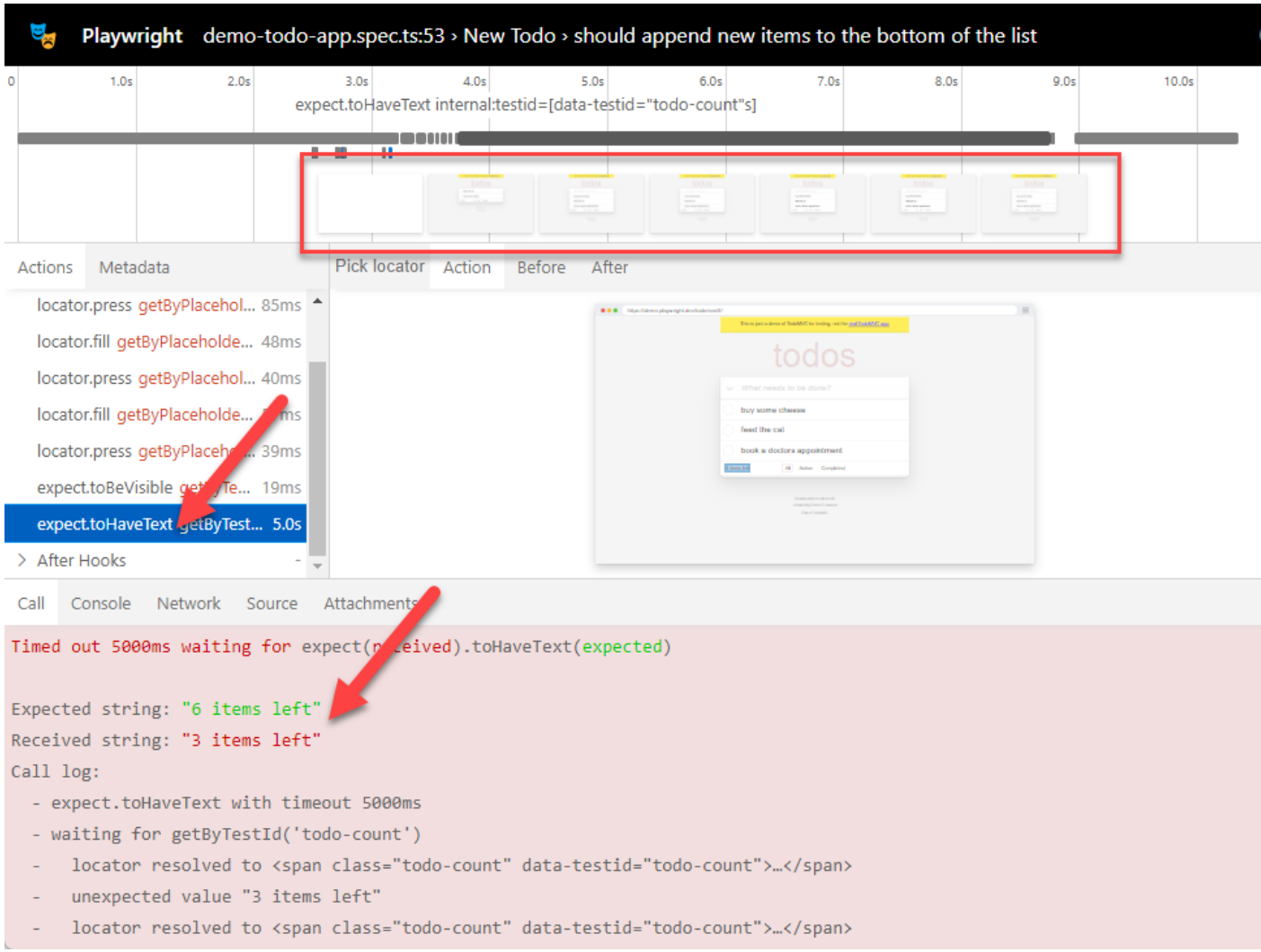
VS Code

- 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**

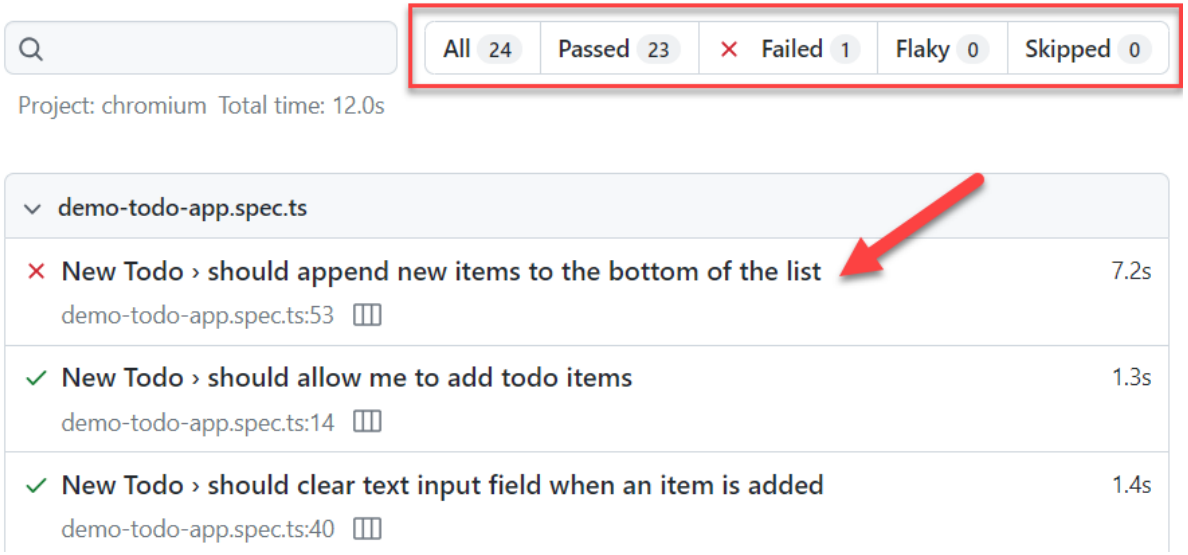
Line Report

- npx playwright test --reporter=**line**

Dot Report

- npx playwright test --reporter=**dot**

1



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 › s
  ✓  4 [firefox] › demo-todo-app.spec.ts:40:7 › New Todo › s
  ✓  5 [chromium] › demo-todo-app.spec.ts:40:7 › New Todo ›
  ✓  6 [firefox] › demo-todo-app.spec.ts:53:7 › New Todo › s

  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

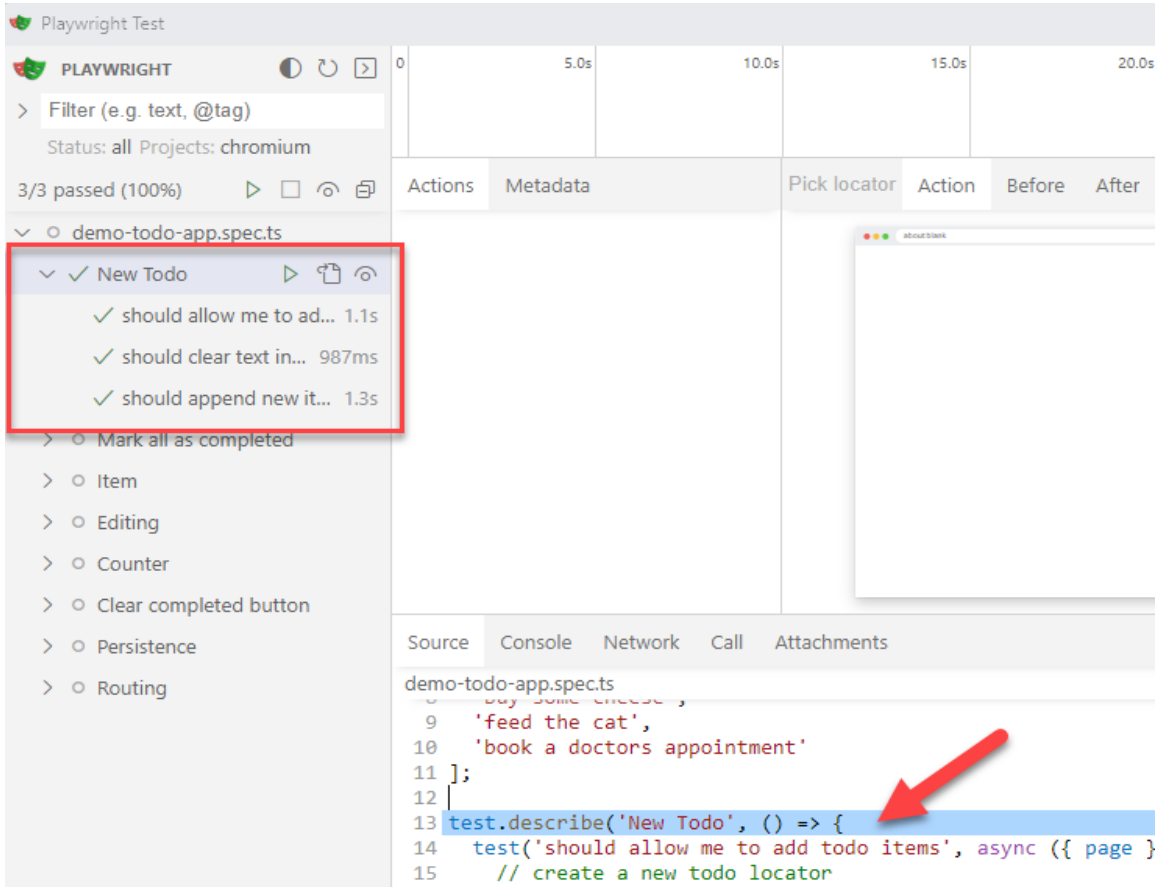
- Trigger events, clicks, real-time page responses.

Element Highlighting

- Hover over elements to highlight them.

Source Code Access

- View HTML, CSS, JS source for debugging.



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!

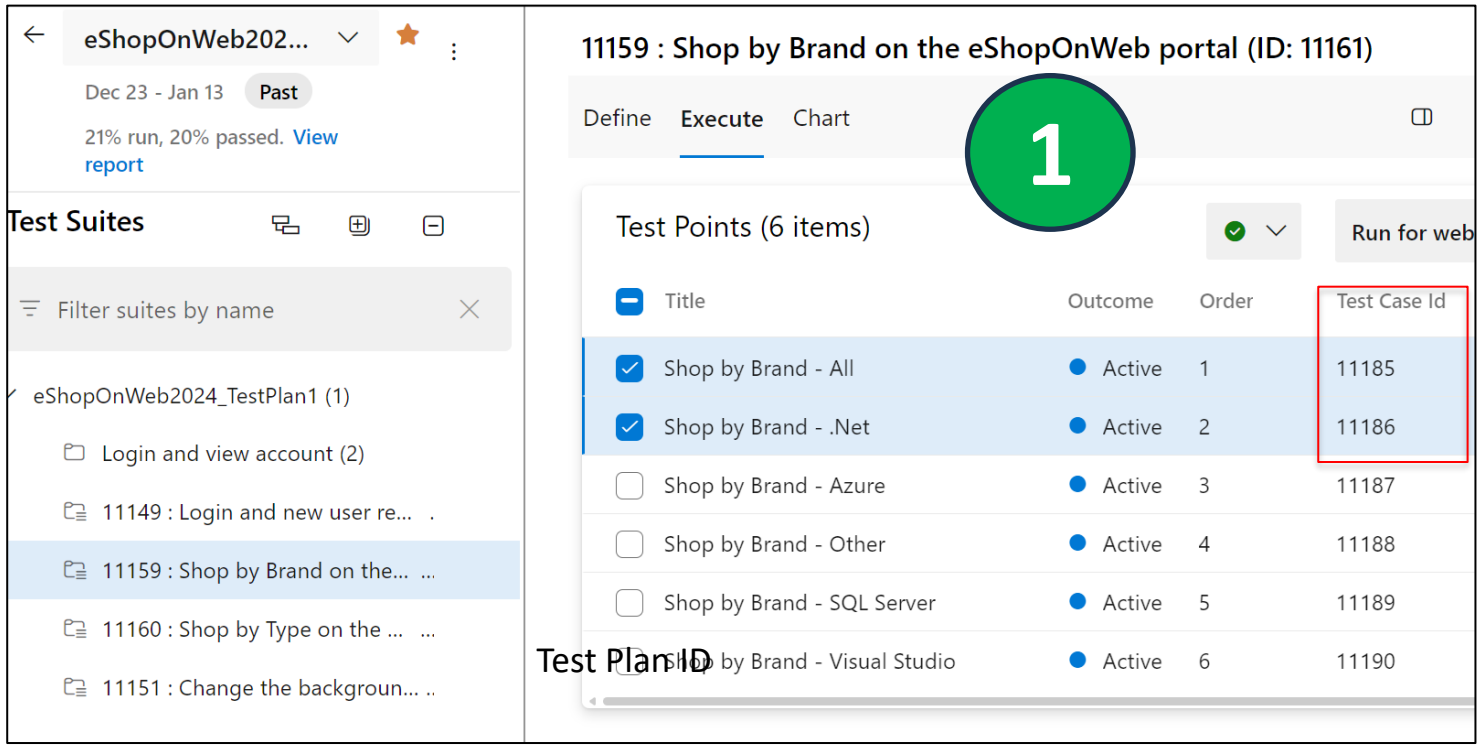
- **End-to-End Traceability**: Requirements (PBI's, User Stories) in Azure Boards are linked to test cases.
- **User-Friendly Testing**: Pipelines enabling the triggering of automated tests through Test Cases, rather than pipelines.
- **History Access**: Azure Test Plans offers access to historical test 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



2

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

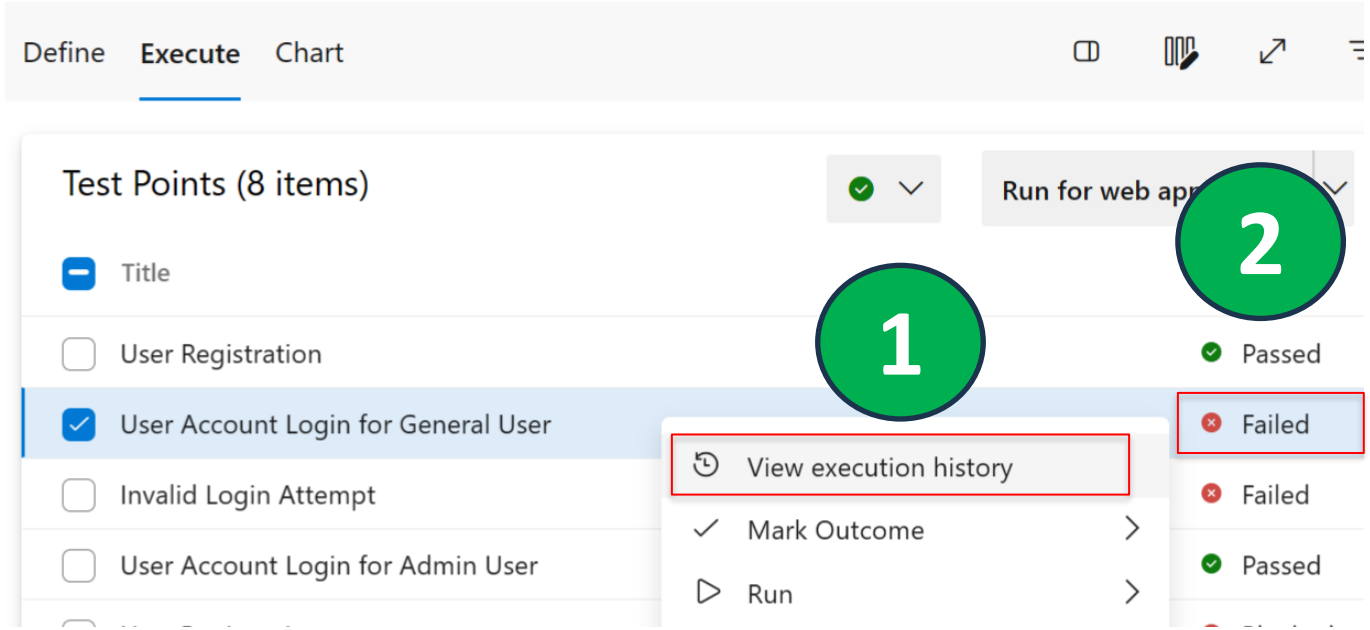
3

```
1   Starting: publish test results
2   ==============================================================================
3   Task         : Publish Test Results
4   Description  : Publish test results to Azure Pipeli
5   Version      : 2.233.1
6   Author       : Microsoft Corporation
7   Help         : https://docs.microsoft.com/azure/dev
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
```

https://marcusfelling.com/

Text Execution History

11149 : Login and new user registraion to the eShopOnWeb portal using email id and password (ID: 11156)



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

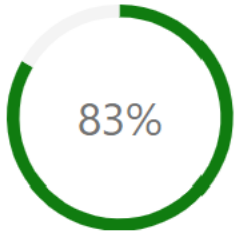
Test Plans Progress Report

Summary

1
Test plans

24
Test points

20 (20 / 24)
Test points run



Run

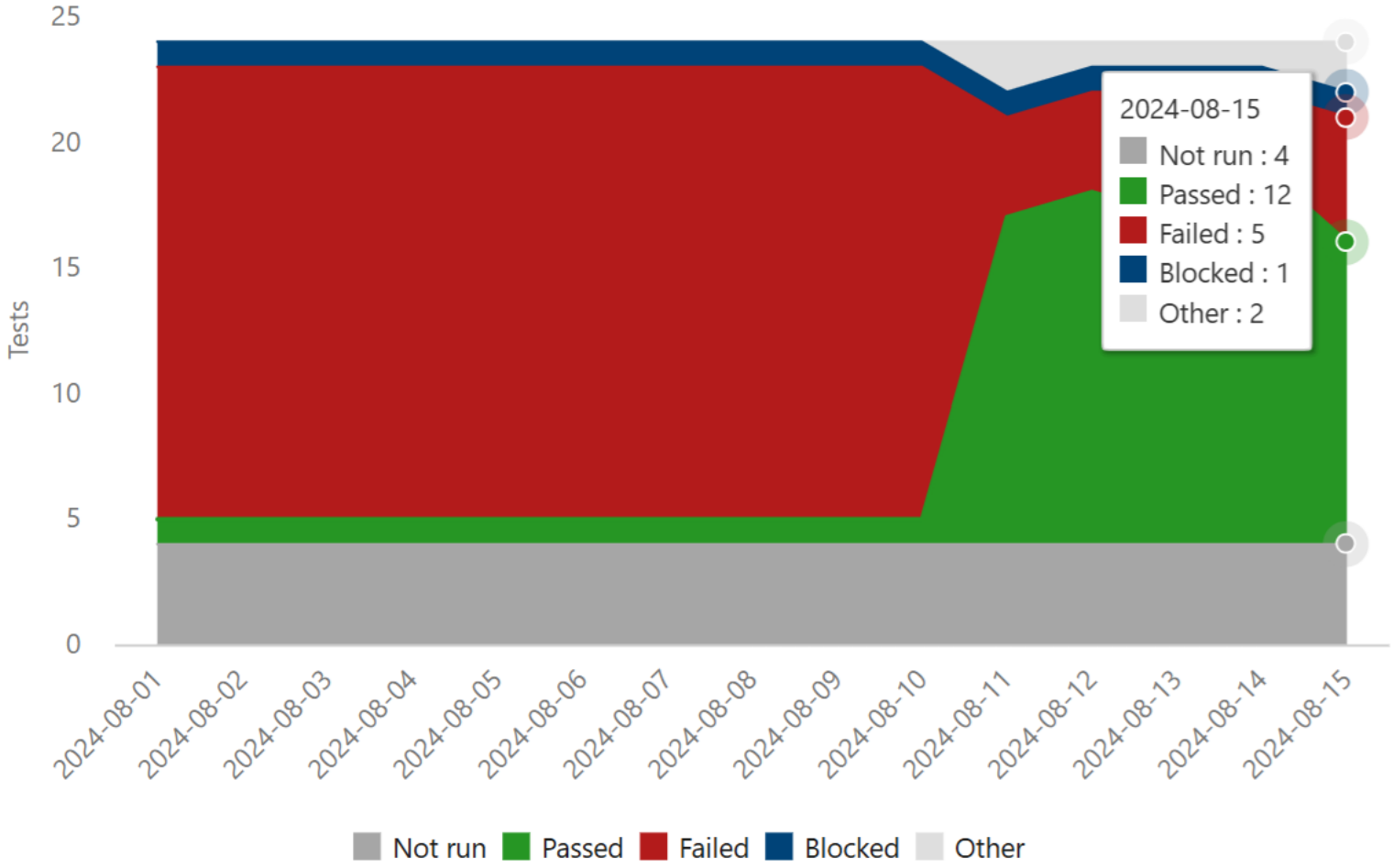
60% (12 / 20)
Pass rate

12 Passed

5 Failed

Outcome trend

Last 14 Days



Demo

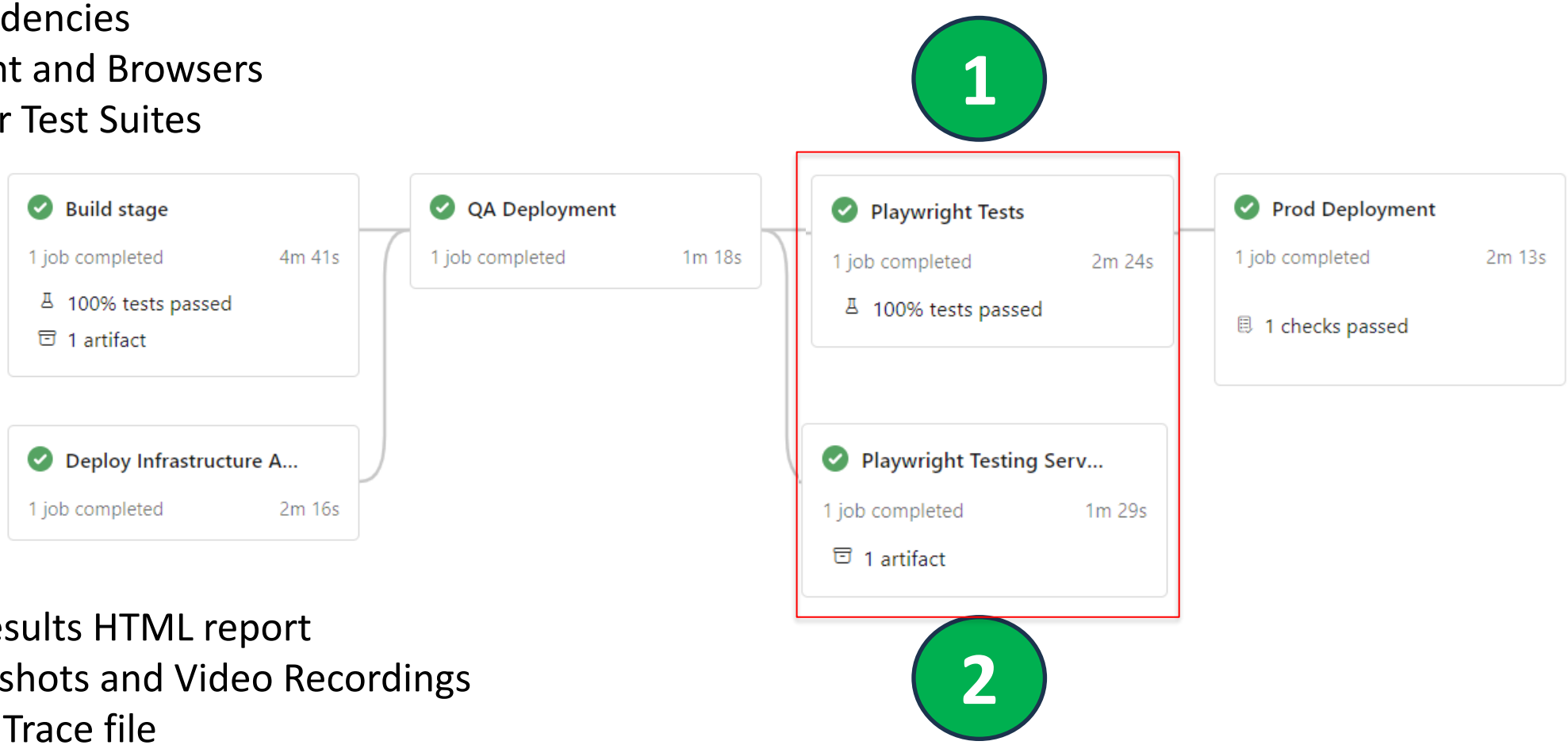
- ✓ Automate Test Cases using Playwright
- ✓ Test Execution History
- ✓ Progress Report

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

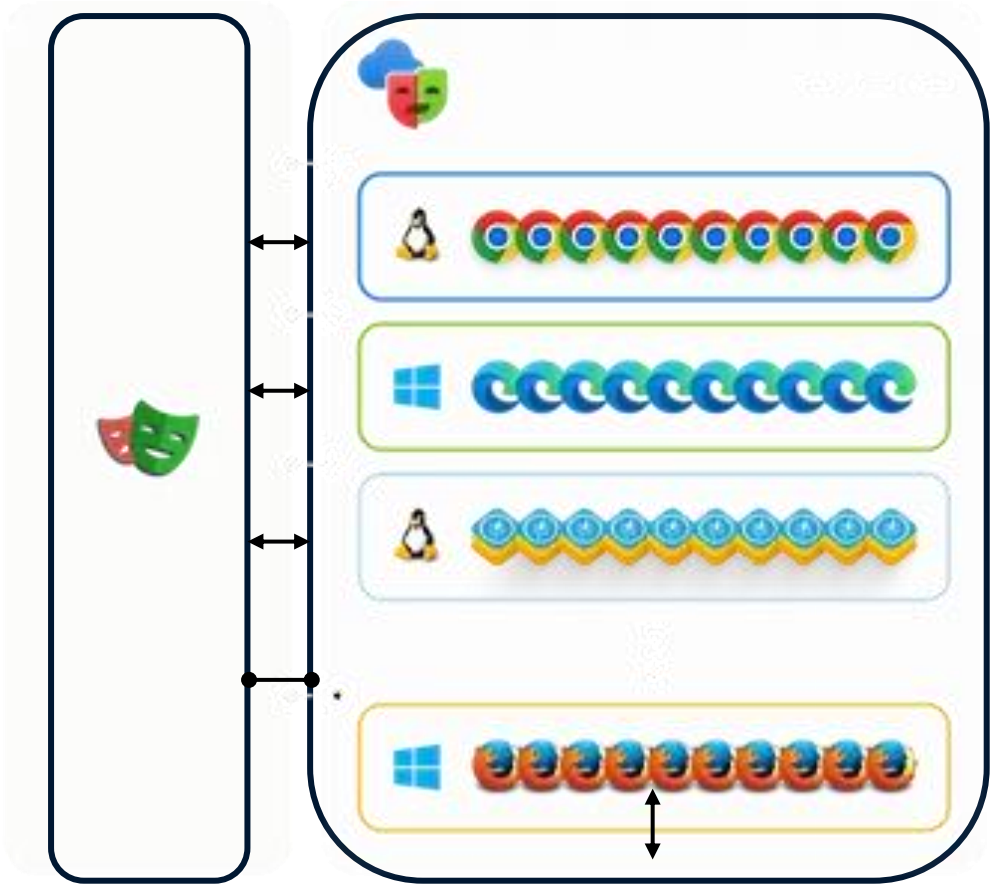


- ✓ Generate Test Results HTML report
- ✓ Generate Screenshots and Video Recordings
- ✓ Generate Debug Trace file
- ✓ Deploy static files to Website or GitHub Pages for viewing

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.



https://azure.microsoft.com/en-us/products/playwright-testing/

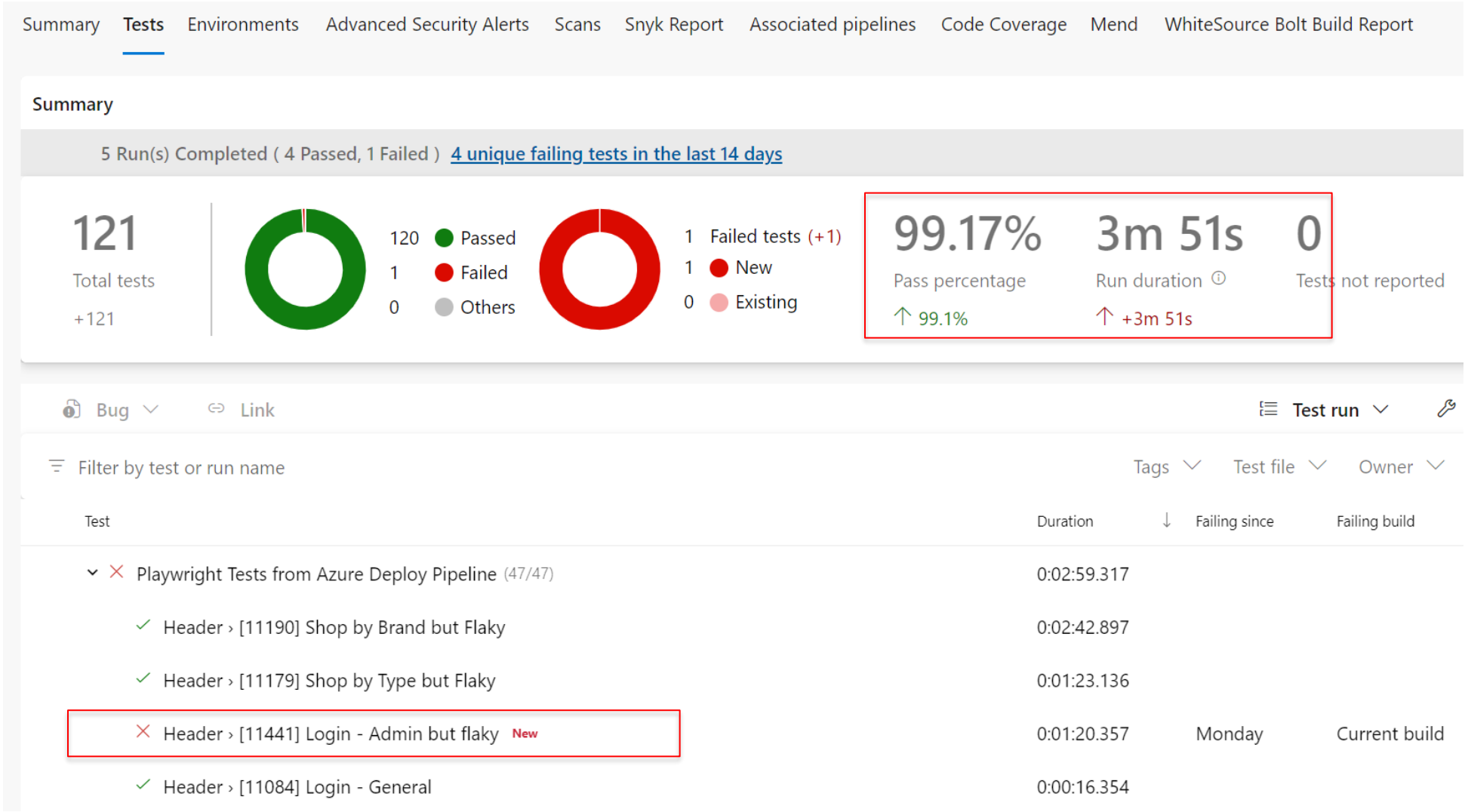
Demo

- ✓ 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

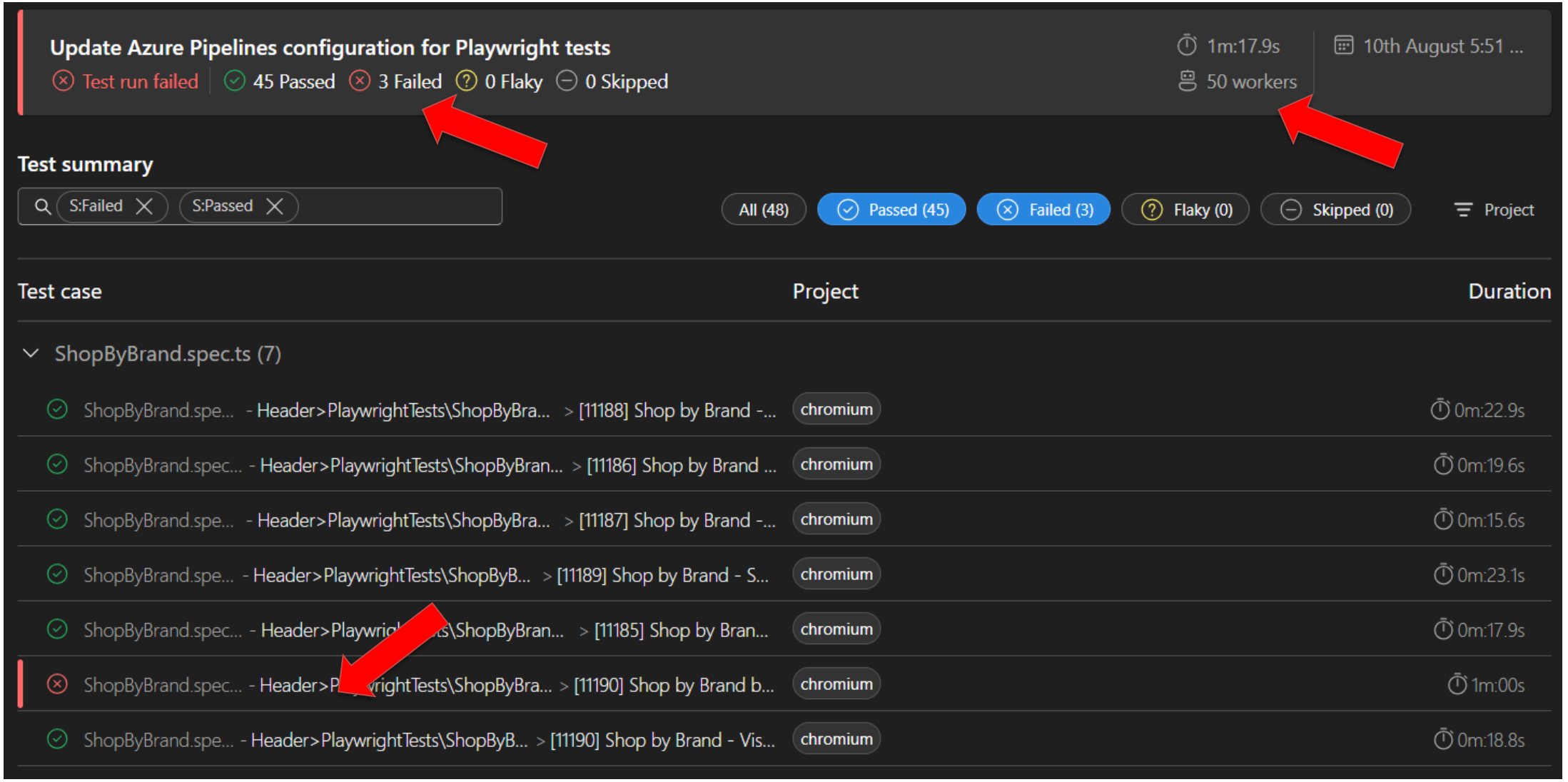
1. Passed!
2. Failed!
3. Percentage!



Microsoft Playwright Testing Rich Reporting

Test results filtered by failed and flaky tests

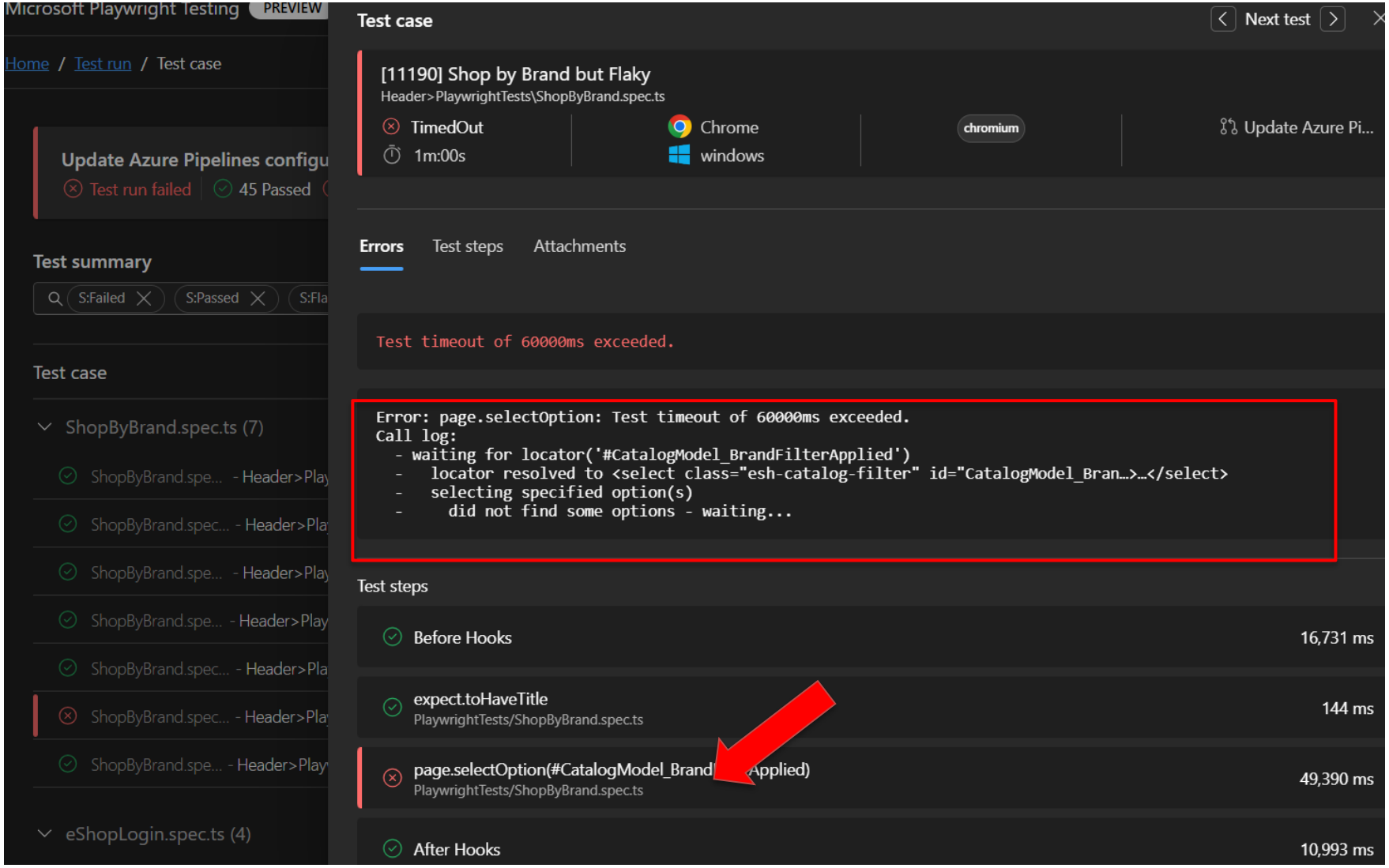
Reporting Private Preview



Microsoft Playwright Testing Rich Reporting

A test result with CI information

Reporting Private Preview



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.getByLabel('User Name').fill(process.env.USERNAME);
  await page.getByLabel('Password').fill(process.env.PASSWORD);
});
```

- 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}}
```

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

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

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

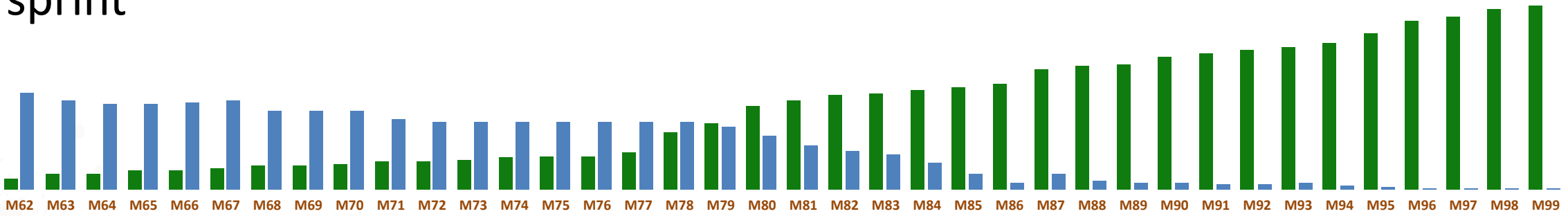
https://automationpanda.com/2023/07/28/modern-web-testing-with-playwright/

Microsoft Azure DevOps team

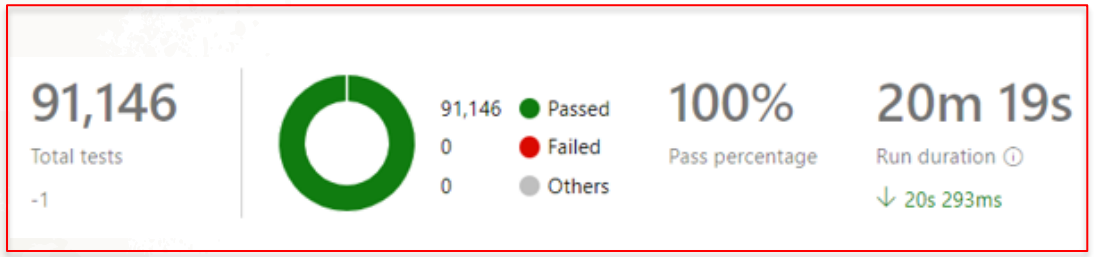
Balancing UI vs Unit Testing in your strategy to maximizes test coverage and effectiveness.

Long running UI functional tests at end of sprint

Shifted to unit tests from automated UI functional tests

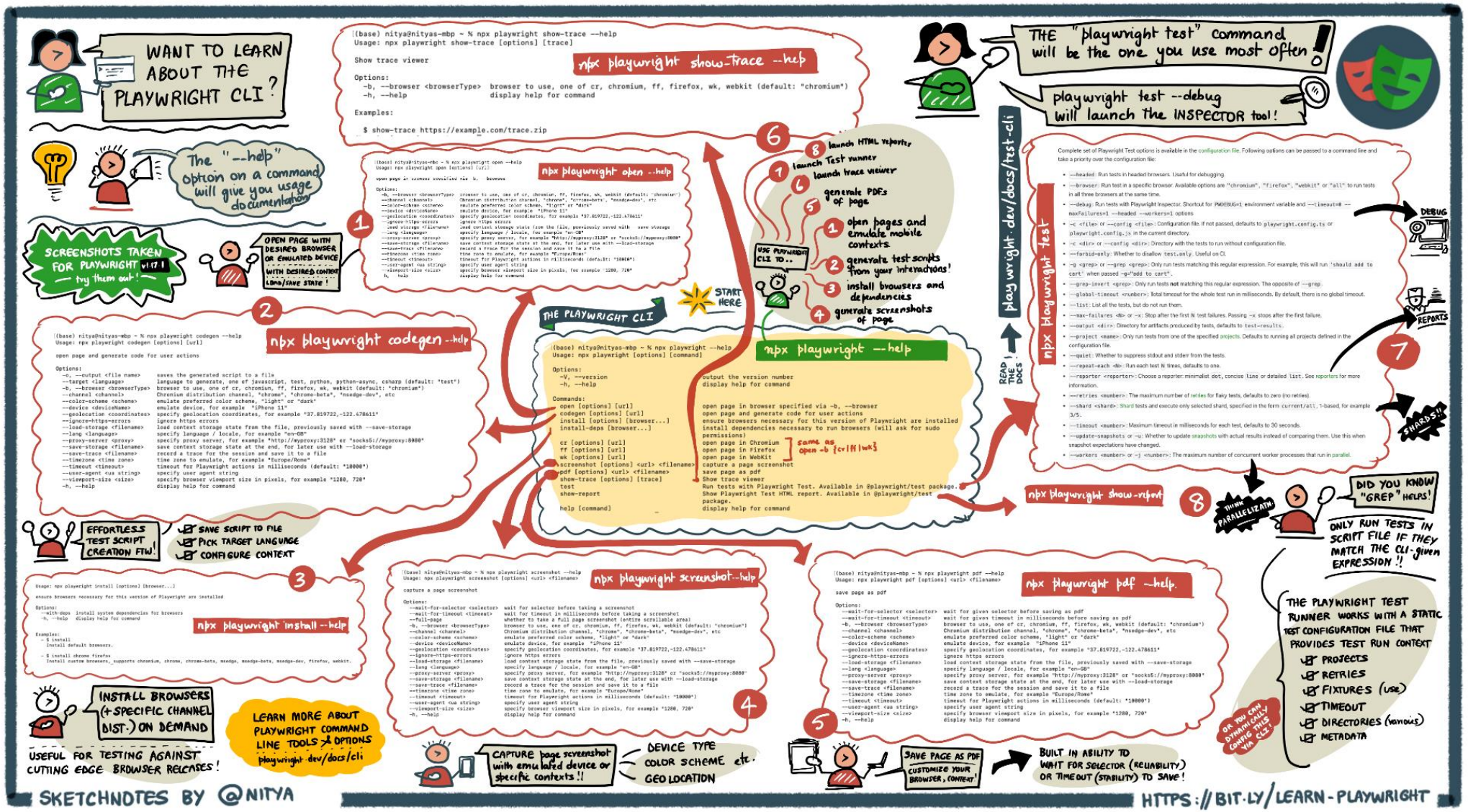


ADO PRs require 100% pass rate before merging



ADO Unit Tests in build pipeline for past 14 days

Playwright CLI



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

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