CI GitHub Actions

Introduction

Playwright tests can be ran on any CI provider. In this section we will cover running tests on GitHub using GitHub actions. If you would like to see how to configure other CI providers check out our detailed doc on Continuous Integration.

To add a GitHub Actions file first create `.github/workflows` folder and inside it add a `playwright.yml` file containing the example code below so that your tests will run on each push and pull request for the main/master branch.

You will learn

- How to run tests on push/pull_request
- How to view test logs
- How to view the trace

Setting up GitHub Actions

On push/pull_request

Tests will run on push or pull request on branches main/master. The workflow will install all dependencies, install Playwright and then run the tests.

.github/workflows/playwright.yml

```
name: Playwright Tests
on:
  push:
    branches: [ main, master ]
  pull_request:
    branches: [ main, master ]
```

```yaml
jobs:
  test:
    timeout-minutes: 60
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v4
    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.11'
    - name: Install dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r requirements.txt
    - name: Ensure browsers are installed
      run: python -m playwright install --with-deps
    - name: Run your tests
      run: pytest --tracing=retain-on-failure
    - uses: actions/upload-artifact@v4
      if: ${{ !cancelled() }}
      with:
        name: playwright-traces
        path: test-results/
```

Via Containers

GitHub Actions support running jobs in a container by using the `jobs.<job_id>.container` option. This is useful to not pollute the host environment with dependencies and to have a consistent environment for e.g. screenshots/visual regression testing across different operating systems.

.github/workflows/playwright.yml

```yaml
name: Playwright Tests
on:
  push:
    branches: [ main, master ]
  pull_request:
    branches: [ main, master ]
jobs:
  playwright:
    name: 'Playwright Tests'
    runs-on: ubuntu-latest
    container:
      image: mcr.microsoft.com/playwright/python:v1.42.0-jammy
    steps:
      - uses: actions/checkout@v4
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: '3.11'
      - name: Install dependencies
        run: |
          python -m pip install --upgrade pip
          pip install -r local-requirements.txt
          pip install -e .
      - name: Run your tests
        run: pytest
        env:
          HOME: /root
```

On deployment

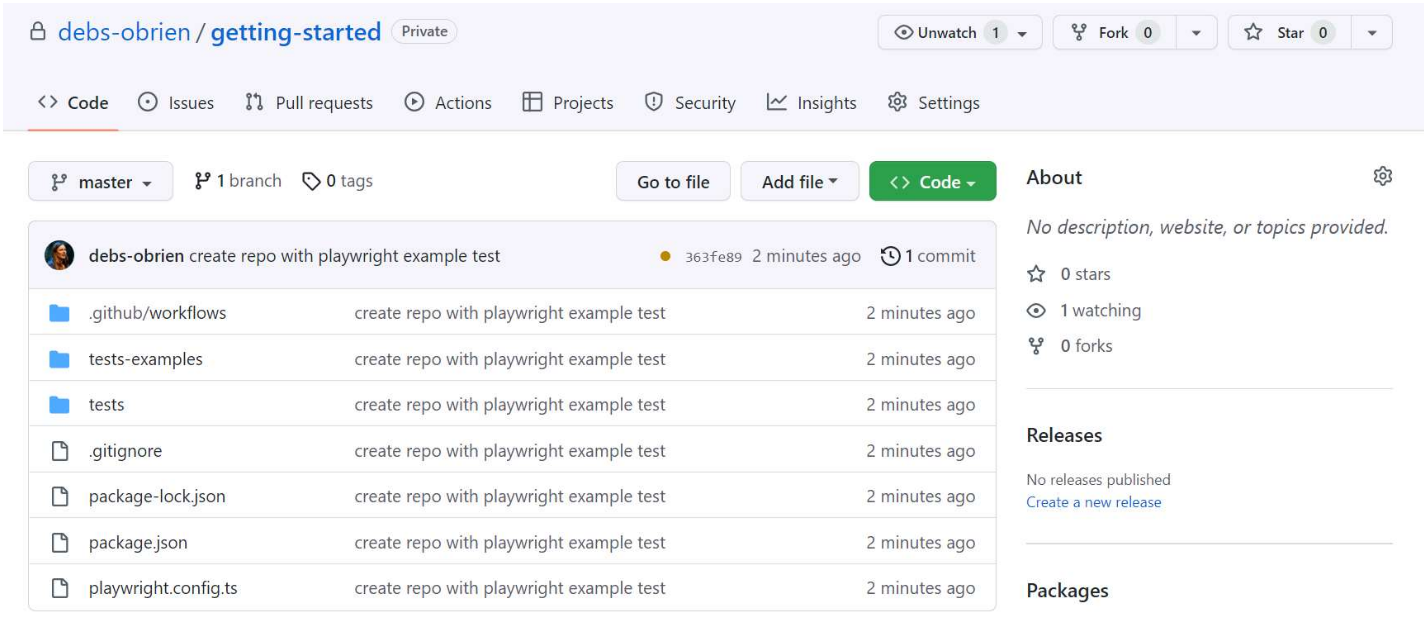
This will start the tests after a GitHub Deployment went into the `success` state. Services like Vercel use this pattern so you can run your end-to-end tests on their deployed environment.

.github/workflows/playwright.yml

```yaml
name: Playwright Tests
on:
  deployment_status:
jobs:
  test:
    timeout-minutes: 60
    runs-on: ubuntu-latest
    if: github.event.deployment_status.state == 'success'
    steps:
    - uses: actions/checkout@v4
      uses: actions/setup-python@v4
      with:
        python-version: '3.11'
    - name: Install dependencies
      run: |
          python -m pip install --upgrade pip
          pip install -r requirements.txt
      - name: Ensure browsers are installed
        run: python -m playwright install --with-deps
      - name: Run tests
        run: pytest
        env:
          # This might depend on your test-runner
          PLAYWRIGHT_TEST_BASE_URL: ${{ github.event.deployment_status.target_url
}}
```

Create a Repo and Push to GitHub

Once you have your GitHub actions workflow setup then all you need to do is Create a repo on GitHub or push your code to an existing repository. Follow the instructions on GitHub and don't forget to initialize a git repository using the `git init` command so you can add, commit and push your code.

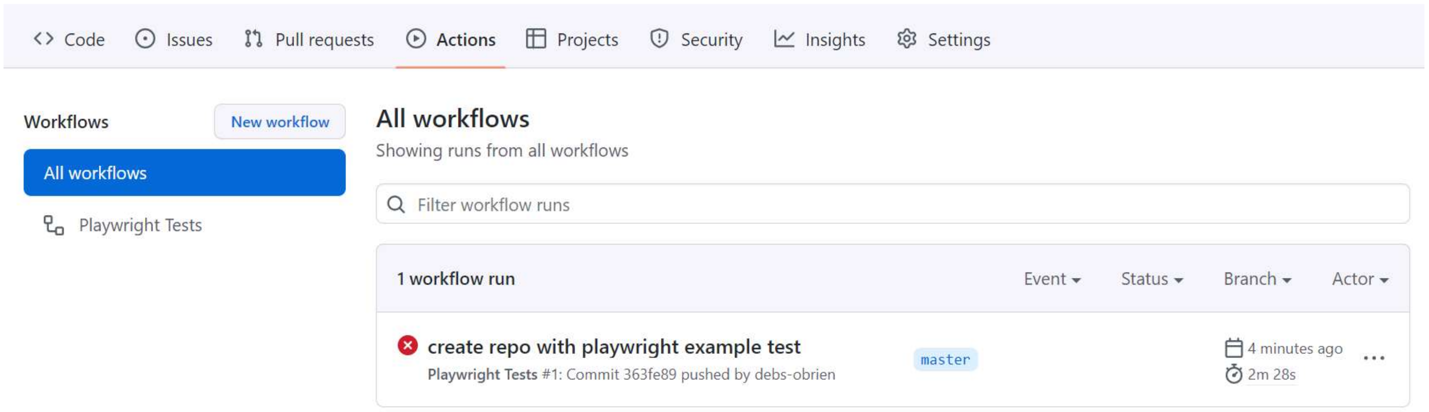


debs-obrien create repo with playwright example test 363fe89 2 minutes ago 1 commit

.github/workflows	create repo with playwright example test	2 minutes ago
tests-examples	create repo with playwright example test	2 minutes ago
tests	create repo with playwright example test	2 minutes ago
.gitignore	create repo with playwright example test	2 minutes ago
package-lock.json	create repo with playwright example test	2 minutes ago
package.json	create repo with playwright example test	2 minutes ago
playwright.config.ts	create repo with playwright example test	2 minutes ago

Opening the Workflows

Click on the **Actions** tab to see the workflows. Here you will see if your tests have passed or failed.

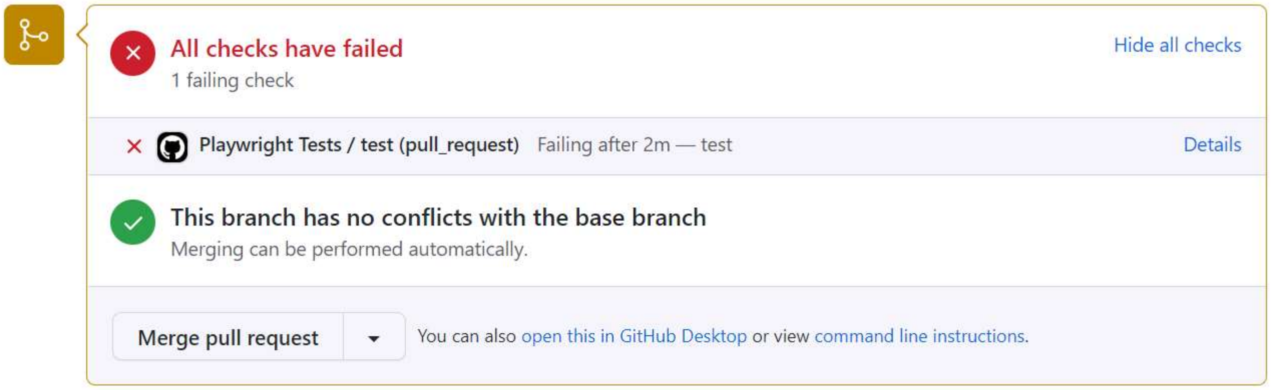


All workflows
Showing runs from all workflows

1 workflow run

create repo with playwright example test
Playwright Tests #1: Commit 363fe89 pushed by debs-obrien
master
4 minutes ago
2m 28s

On Pull Requests you can also click on the **Details** link in the PR status check.



All checks have failed
1 failing check

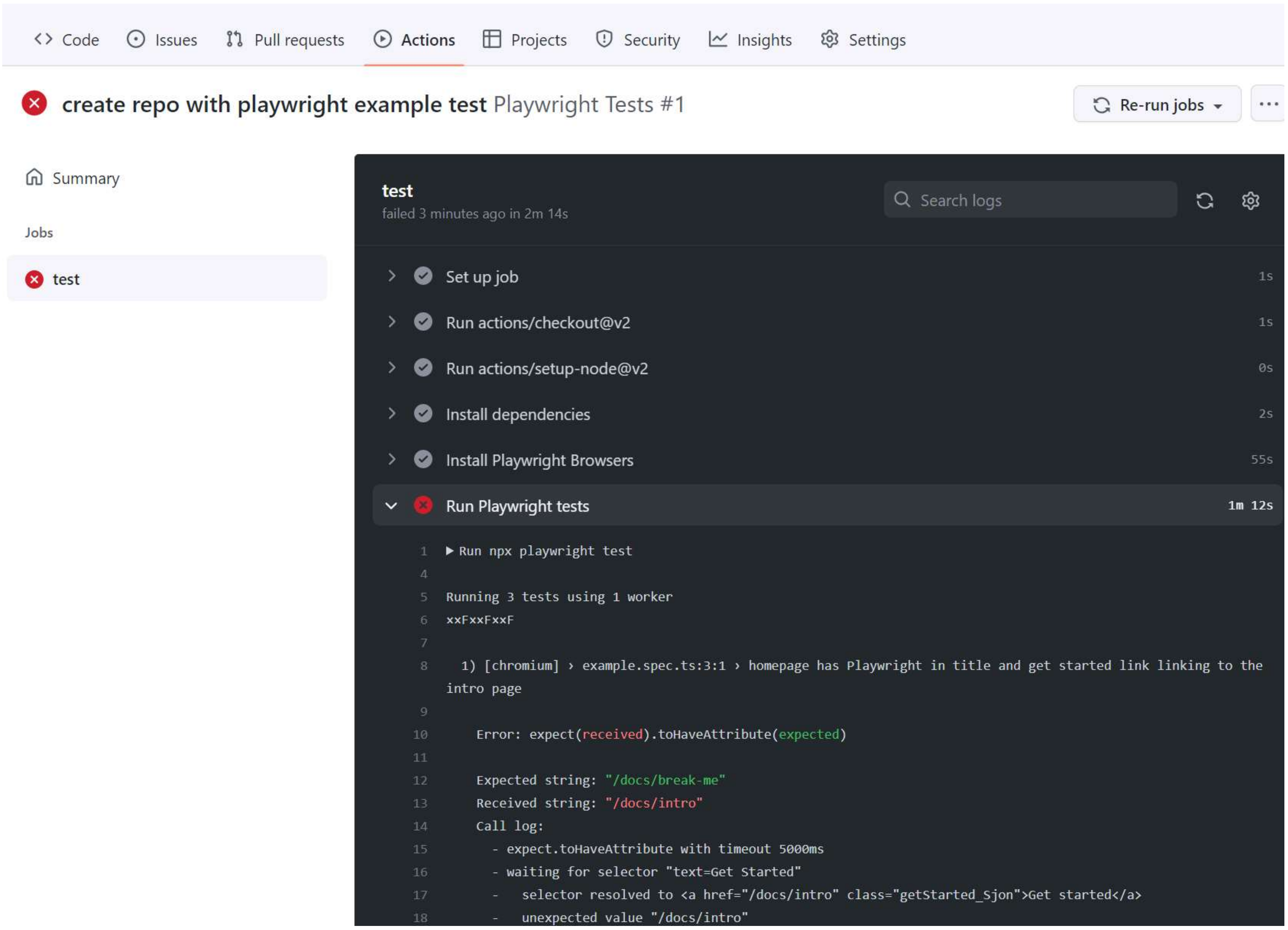
Playwright Tests / test (pull_request) Failing after 2m — test

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

Viewing Test Logs

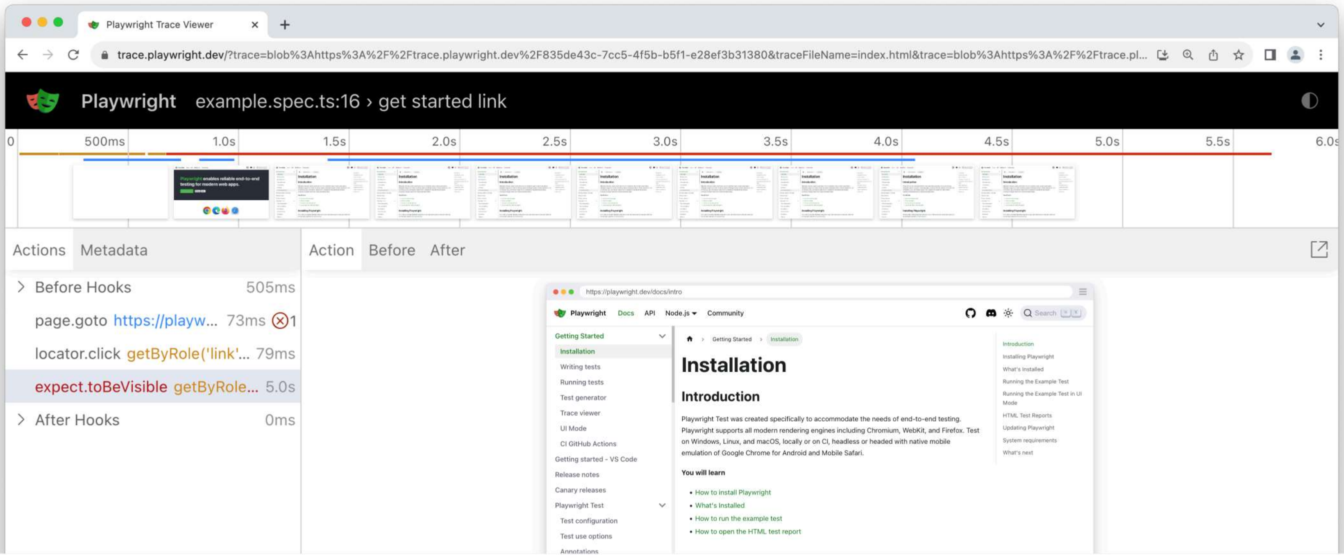
Clicking on the workflow run will show you the all the actions that GitHub performed and clicking on **Run Playwright tests** will show the error messages, what was expected and what was received as well as the call log.


Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
create repo with playwright example test Playwright Tests #1
Re-run jobs
Summary
Jobs
test
test
failed 3 minutes ago in 2m 14s
Search logs
Set up job 1s
Run actions/checkout@v2 1s
Run actions/setup-node@v2 0s
Install dependencies 2s
Install Playwright Browsers 55s
Run Playwright tests 1m 12s

```
▶ Run npx playwright test

Running 3 tests using 1 worker
xxFxxFxxF

  1) [chromium] › example.spec.ts:3:1 › homepage has Playwright in title and get started link linking to the
intro page

    Error: expect(received).toHaveAttribute(expected)

    Expected string: "/docs/break-me"
    Received string: "/docs/intro"
    Call log:
      - expect.toHaveAttribute with timeout 5000ms
      - waiting for selector "text=Get Started"
      -   selector resolved to <a href="/docs/intro" class="getStarted_Sjon">Get started</a>
      -   unexpected value "/docs/intro"
```

Viewing the Trace

trace.playwright.dev is a statically hosted variant of the Trace Viewer. You can upload trace files using drag and drop.


Playwright Trace Viewer
Playwright example.spec.ts:16 › get started link
Actions
Metadata
Before Hooks 505ms
page.goto https://playw... 73ms 1
locator.click getByRole('link'... 79ms
expect.toBeVisible getByRole... 5.0s
After Hooks 0ms
Action
Before
After
Installation
Introduction

What's Next

- Learn how to use Locators
- Learn how to perform Actions
- Learn how to write Assertions
- Learn more about the Trace Viewer
- Learn more about running tests on other CI providers