RISHIKESH VAJRE

Playwright + GitHub Actions

This guide provides a single, consistent approach to setting up Playwright with GitHub Actions. It includes files, exact commands, and notes that match the configuration and workflow. Simply copy and paste the file contents as instructed.

Consistent Setup

A unified method for Playwright and GitHub Actions.

Exact Resources

All necessary files and precise commands provided.

Matching Workflow

Configuration and workflow designed to align perfectly.

Simple Implementation

Copy and paste for quick and easy setup.

Deploy to GitHub

1 Initialize Git Repository

```
git init
```

2 Stage All Files

```
git add .
```

3 Create Initial Commit

```
git commit -m "Playwright + GitHub Actions demo"
```

4 Set Main Branch

```
git branch -M main
```

5 Add Remote Origin

```
git remote add origin [your-repo-url]
```

6 Push to GitHub

```
git push -u origin main
```

Then open GitHub → Actions → select the workflow run to see logs and artifacts.

Project Setup & Installation

01

Create project folder

```
mkdir playwright-github-actions-demo
cd playwright-github-actions-demo
```

02

Initialize npm & install Playwright

```
npm init -y
npm i -D @playwright/latest
```

03

Install browsers (important!)

```
npx playwright install --with-deps
```

Package.json Configuration

Create or replace your package.json scripts section with this (full file shown for clarity):

```
{
 "name": "playwright-github-actions-demo",
 "version": "1.0.0",
 "scripts": {
 "test": "npx playwright test",
 "test:headed": "npx playwright test --headed",
 "show-report": "npx playwright show-report"
 },
 "devDependencies": {
 "@playwright/test": "^1.48.0"
 }
}
```

Playwright Configuration

Create `playwright.config.ts` with these production-ready contents:

```
import { defineConfig, devices } from '@playwright/test';

export default defineConfig({
  testDir: './tests',

  // Per-test timeout (ms)
  timeout: 60_000,

// Expect assertions timeout
expect: { timeout: 5_000 },

// Run test files in parallel when possible
fullyParallel: true,

// Fail the run if test.only is left in the source (useful on CI)
forbidOnly: !!process.env.CI,

// Retries/workers: more retries on CI
retries: process.env.CI ? 2 : 0,
workers: process.env.CI ? 2 : undefined,

// Reporters: GitHub Actions friendly + junit + html
reporter: process.env.CI
  ? [['github'], ['junit', { outputFile: 'results/junit/results.xml' }], ['html', { open: 'never' }]]
  : [['list'], ['html', { open: 'never' }]],

// Where to store screenshots/traces/videos
outputDir: 'test-results/',

use: {
  baseURL: process.env.BASE_URL ?? 'https://playwright.dev',
  // Action/navigation timeouts
  actionTimeout: 0, // single action unlimited (adjust if you want)
  navigationTimeout: 30_000, // navigation timeout in ms

  // Helpful artifact settings
  trace: 'on-first-retry',
  screenshot: 'only-on-failure',
  video: 'retain-on-failure',
  // Misc
  ignoreHTTPSErrors: true,
  headless: true,
  viewport: { width: 1280, height: 720 },
},

projects: [
  { name: 'chromium', use: { ...devices['Desktop Chrome'] } },
  { name: 'firefox', use: { ...devices['Desktop Firefox'] } },
  { name: 'webkit', use: { ...devices['Desktop Safari'] } },
],

// If you test a local app, this will start it for you (adjust as needed)
 webServer: {
   command: 'npm run start',
   url: 'http://localhost:3000',
   reuseExistingServer: !process.env.CI,
 },
});
```

Key Features

- baseURL pulls from BASE_URL env var (useful for local/staging/prod)
- webServer will run npm run start before tests — add a start script if you use this
- Artifacts (html report → default playwright-report) + junit path results/junit/results.xml

Example Test File

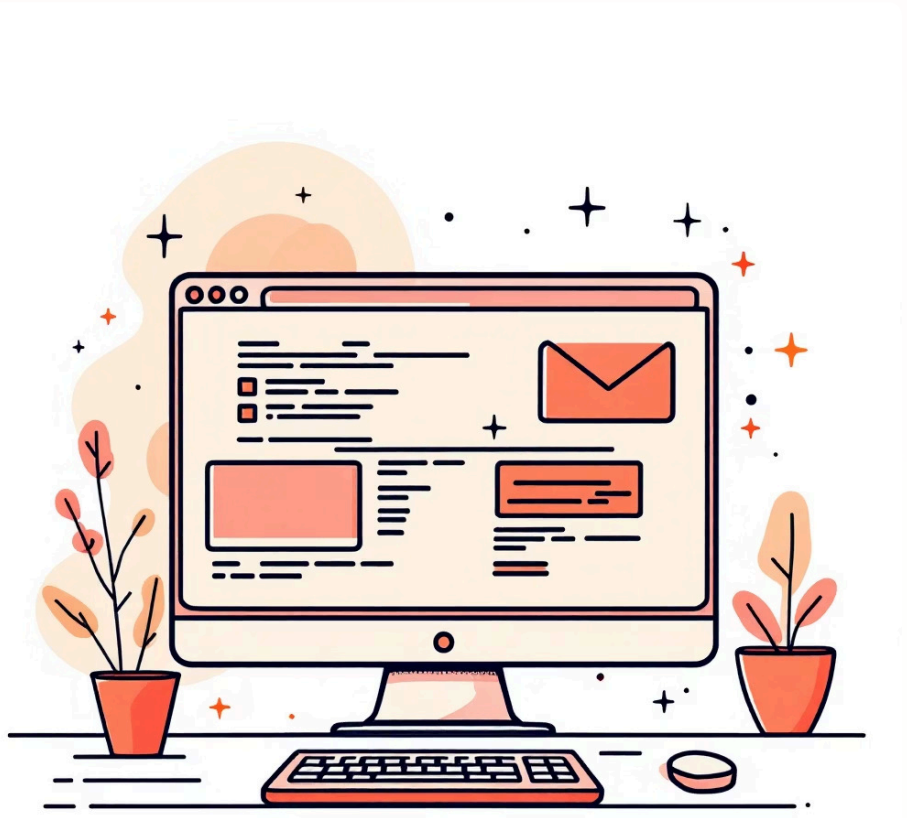
Create `tests/example.spec.ts`:

```
import { test, expect } from '@playwright/test';

test('homepage has Playwright in title and get started link', async ({ page }) => {
  await page.goto('/');
  await expect(page).toHaveTitle(/Playwright/);

  const getStarted = page.getByRole('link', { name: 'Get started' });
  await expect(getStarted).toBeVisible();
  await getStarted.click();

  await expect(page).toHaveURL(/.*intro/);
});
```

Test Structure

This example test demonstrates the core Playwright testing patterns:

- Page navigation with `page.goto()`
- Title assertions using regex patterns
- Element interaction with role-based selectors
- URL validation after navigation



GitHub Actions Workflow

Create `.github/workflows/playwright.yml` with the commented workflow below:

yaml

Copy code

```yaml
# .github/workflows/deploy-playwright-report.yml
name: Deploy Playwright Report to Pages

on:
  workflow_dispatch:
  push:
    branches: [ main ]

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: 'npm'

      - name: Install dependencies
        run: npm install

      - name: Install browsers
        run: npx playwright install --with-deps

      - name: Run tests (generate report)
        run: npm test

      - name: Deploy to gh-pages
        uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./playwright-re
```

Key Points

- Actions will automatically set CI=true
- `playwright-report` is uploaded as an artifact even on failure so you can debug

Git Configuration

Create `.gitignore` to avoid committing artifacts:

```
node_modules/
.playwright/
playwright-report/
test-results/
results/
.vscode/
```

Why These Exclusions?

These directories contain generated files, dependencies, and artifacts that shouldn't be tracked in version control. They can be recreated during the build process.

Local Development Commands

1

Install Dependencies

```
npm install
```

Install project dependencies if you haven't already

2

Install Browsers

```
npx playwright install --with-deps
```

Download required browser binaries and system dependencies

3

Run All Tests

```
npm test
```

Execute the complete test suite

Advanced Commands

Targeted Testing

```
npx playwright test tests/example.spec.ts -p chromium
```

Run single test file or specific project

Custom Base URL

```
BASE_URL="https://staging.example.com" npm test
```

Run tests pointing to an external base URL

View Reports

```
npx playwright show-report playwright-report
```

View HTML report locally after a run

Start Development Server

```
npm run start
```

If using webServer in config and your app runs on port 3000

Optional: GitHub Pages Deployment

If you want the Playwright HTML report viewable online, add a second workflow that runs tests and deploys `playwright-report` to GitHub Pages. Example (simplified):

```
# .github/workflows/deploy-playwright-report.yml
name: Deploy Playwright Report to Pages

on:
  workflow_dispatch:
  push:
    branches: [ main ]

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4

      - name: Setup Node
        uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: 'npm'
      - name: Install dependencies
        run: npm install
      - name: Install browsers
        run: npx playwright install --with-deps
      - name: Run tests (generate report)
        run: npm test
      - name: Deploy to gh-pages
        uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./playwright-report
```

Configuration

In your repository Settings → Pages, choose the gh-pages branch as the source (or let the action create it and configure)

Public Access

This makes the report publicly viewable (or limited by page settings)

Troubleshooting & Quick Tips

CI Test Failures

If tests fail on CI but pass locally: download `playwright-report` artifact in Actions and open it with `npx playwright show-report path` to inspect traces/screenshots.

Flaky Tests

Use `trace: 'on-first-retry'` and retries (we set retries on CI) to help debug flakes.

WebServer Timeouts

If webServer timeouts: ensure `npm run start` responds on http://localhost:3000 or change url.

Test.only Prevention

If you accidentally left `test.only`, `forbidOnly: !!process.env.CI` will fail CI so you don't silently skip tests.

Resource Management

Tune workers in CI to the GitHub runner capacity if tests are resource heavy.

Success! You now have a complete Playwright + GitHub Actions setup that's production-ready with proper error handling, artifact collection, and debugging capabilities.