GitHub Actions CI/CD Setup Guide for Playwright

> **Complete Step-by-Step Guide: From Zero to Automated Testing Pipeline**

Table of Contents

1. Prerequisites
2. Understanding GitHub Actions
3. Step-by-Step Setup
4. Workflow File Explained
5. Running Your First Pipeline
6. Viewing Test Results & Reports
7. Advanced Configurations
8. Troubleshooting
9. Best Practices

1. Prerequisites

Before setting up CI/CD, ensure you have:

Requirement	Description	How to Verify
GitHub Account	Free account at github.com	Login to github.com
Git Installed	Version control on local machine	`git --version`
Repository Created	Your Playwright project in a GitHub repo	Check github.com/your-username
Playwright Tests	Working tests locally	`npx playwright test` passes

2. Understanding GitHub Actions

What is GitHub Actions?

GitHub Actions is a CI/CD platform built into GitHub that allows you to automate your software workflows directly in your repository.

Key Concepts

```
┌─────────────────────────────────────────────────────────────┐
│                     GITHUB ACTIONS                          │
├─────────────────────────────────────────────────────────────┤
│  WORKFLOW (.yml file)                                       │
│  └── Automated process defined in your repository          │
│                                                             │
│  EVENT (trigger)                                            │
│  └── What starts the workflow (push, pull_request, etc.)   │
│                                                             │
│  JOB                                                        │
│  └── Set of steps that execute on the same runner          │
│                                                             │
│  STEP                                                       │
│  └── Individual task within a job                          │
│                                                             │
│  ACTION                                                     │
│  └── Reusable unit of code (e.g., actions/checkout)        │
│                                                             │
│  RUNNER                                                     │
│  └── Server that runs your workflows (ubuntu, windows)     │
└─────────────────────────────────────────────────────────────┘
```

Workflow Triggers

Trigger	Description	Example
`push`	Code pushed to branch	Push to main/develop
`pull_request`	PR opened/updated	PR to main branch
`schedule`	Cron-based schedule	Daily at midnight

Trigger	Description	Example
`workflow_dispatch`	Manual trigger	Run from GitHub UI

3. Step-by-Step Setup

Step 1: Initialize Git Repository (If Not Done)

```
# Navigate to your project
cd "d:\Web Automation Testing Playground playwright"

# Initialize git (skip if already initialized)
git init

# Add all files
git add .

# Create initial commit
git commit -m "Initial commit: Playwright automation framework"
```

Step 2: Create GitHub Repository

1. Go to github.com
2. Click **"+"** → **"New repository"**
3. Enter repository name: `playwright-automation-framework`
4. Choose **Public** or **Private**
5. Click **"Create repository"**

Step 3: Connect Local to Remote

```
# Add remote origin (replace with your URL)
git remote add origin https://github.com/YOUR_USERNAME/playwright-automation-framework.git

# Push to GitHub
git branch -M main
git push -u origin main
```

Step 4: Create Workflow Directory

```bash
# Create the required directory structure
mkdir -p .github/workflows
```

Directory Structure:

```
your-project/
├── .github/
│   └── workflows/
│       └── playwright.yml    ← Workflow file goes here
├── tests/
├── pages/
└── ...
```

Step 5: Create Workflow File

Create `.github/workflows/playwright.yml` with the following content:

```yaml
name: Playwright Tests

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main]
  workflow_dispatch:

jobs:
  test:
    timeout-minutes: 60
    runs-on: ubuntu-latest

    steps:
      - name: Checkout repository
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 20

      - name: Install dependencies
        run: npm ci

      - name: Install Playwright Browsers
        run: npx playwright install --with-deps

      - name: Run Playwright tests
        run: npx playwright test

      - name: Upload test report
        uses: actions/upload-artifact@v4
        if: always()
        with:
          name: playwright-report
          path: playwright-report/
          retention-days: 30
```

Step 6: Commit and Push Workflow

```
# Add the workflow file
git add .github/workflows/playwright.yml

# Commit
git commit -m "Add GitHub Actions CI/CD workflow for Playwright tests"

# Push to trigger the workflow
git push origin main
```

Step 7: Verify Workflow Running

1. Go to your GitHub repository
2. Click **"Actions"** tab
3. You should see your workflow running!

4. Workflow File Explained

Complete Annotated Workflow

```yaml
# ==================================================
# WORKFLOW NAME - Displayed in GitHub Actions tab
# ==================================================
name: Playwright Tests

# ==================================================
# TRIGGERS - When should this workflow run?
# ==================================================
on:
  # Run on push to these branches
  push:
    branches: [main, develop]

  # Run on pull requests to main
  pull_request:
    branches: [main]

  # Allow manual trigger from GitHub UI
  workflow_dispatch:

# ==================================================
# JOBS - What work should be done?
# ==================================================
jobs:
  # Job ID (can be any name)
  test:
    # Maximum time for job to run
    timeout-minutes: 60

    # Operating system for the runner
    runs-on: ubuntu-latest

    # ============================================
    # STEPS - Individual tasks in sequence
    # ============================================
    steps:
      # Step 1: Get your code
      - name: Checkout repository
```

```yaml
        uses: actions/checkout@v4

      # Step 2: Set up Node.js environment
      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: 20

      # Step 3: Install npm packages
      - name: Install dependencies
        run: npm ci

      # Step 4: Install browsers for Playwright
      - name: Install Playwright Browsers
        run: npx playwright install --with-deps

      # Step 5: Execute tests
      - name: Run Playwright tests
        run: npx playwright test

      # Step 6: Save test reports as artifacts
      - name: Upload test report
        uses: actions/upload-artifact@v4
        if: always()  # Upload even if tests fail
        with:
          name: playwright-report
          path: playwright-report/
          retention-days: 30
```

Key Actions Explained

Action	Purpose
`actions/checkout@v4`	Clones your repository code
`actions/setup-node@v4`	Installs Node.js on the runner
`actions/upload-artifact@v4`	Saves files for download later

5. Running Your First Pipeline

Automatic Triggers

The workflow runs automatically when you:

- Push code to `main` or `develop` branch
- Create/update a pull request to `main`

Manual Trigger

1. Go to **Actions** tab
2. Select **"Playwright Tests"** workflow
3. Click **"Run workflow"** dropdown
4. Select branch and click **"Run workflow"**

Viewing Pipeline Progress

```
┌─────────────────────────────────────────────────────────────┐
│  GitHub Repository → Actions Tab                            │
├─────────────────────────────────────────────────────────────┤
│                                                             │
│  ● Playwright Tests                                         │
│    └── Run #1: Commit message here                          │
│        ├── ✓ Checkout repository (2s)                       │
│        ├── ✓ Setup Node.js (5s)                             │
│        ├── ✓ Install dependencies (30s)                     │
│        ├── ✓ Install Playwright Browsers (60s)              │
│        ├── ● Run Playwright tests (running...)              │
│        └── ○ Upload test report (waiting)                   │
│                                                             │
└─────────────────────────────────────────────────────────────┘
```

6. Viewing Test Results & Reports

Method 1: View Console Output

1. Click on the running/completed workflow
2. Click on the **"test"** job
3. Expand **"Run Playwright tests"** step
4. View test output in console

Method 2: Download HTML Report

1. After workflow completes, go to workflow run
2. Scroll to **"Artifacts"** section at bottom
3. Click **"playwright-report"** to download
4. Extract ZIP and open `index.html`

Method 3: GitHub Pages (Advanced)

Deploy reports to GitHub Pages for permanent access:

```yaml
    - name: Deploy report to GitHub Pages
      uses: peaceiris/actions-gh-pages@v3
      if: always()
      with:
        github_token: ${{ secrets.GITHUB_TOKEN }}
        publish_dir: ./playwright-report
```

7. Advanced Configurations

Multi-Browser Testing with Matrix

```yaml
jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      fail-fast: false
      matrix:
        browser: [chromium, firefox, webkit]

    steps:
      # ... checkout, setup steps ...

      - name: Run Playwright tests
        run: npx playwright test --project=${{ matrix.browser }}
```

Sharding (Split Tests Across Runners)

```yaml
jobs:
  test:
    runs-on: ubuntu-latest
    strategy:
      fail-fast: false
      matrix:
        shard: [1/4, 2/4, 3/4, 4/4]

    steps:
      # ... setup steps ...

      - name: Run Playwright tests
        run: npx playwright test --shard=${{ matrix.shard }}
```

Scheduled Runs (Daily/Weekly)

```yaml
on:
  schedule:
    # Run every day at midnight UTC
    - cron: '0 0 * * *'

    # Run every Monday at 9 AM UTC
    - cron: '0 9 * * 1'
```

Environment Variables & Secrets

```yaml
jobs:
  test:
    runs-on: ubuntu-latest
    env:
      BASE_URL: https://staging.example.com

    steps:
      - name: Run tests
        run: npx playwright test
        env:
          TEST_USER: ${{ secrets.TEST_USERNAME }}
          TEST_PASS: ${{ secrets.TEST_PASSWORD }}
```

Setting Secrets:

1. Go to repository **Settings**
2. Click **Secrets and variables** → **Actions**
3. Click **New repository secret**
4. Add name and value

Cross-Platform Testing

```yaml
jobs:
  test:
    strategy:
      matrix:
        os: [ubuntu-latest, windows-latest, macos-latest]
    runs-on: ${{ matrix.os }}
```

8. Troubleshooting

Common Issues & Solutions

Issue	Solution
Browser install fails	Use `npx playwright install --with-deps`
Tests timeout	Increase `timeout-minutes` in job
npm ci fails	Ensure `package-lock.json` is committed
Permission denied	Check file permissions in repository
Out of disk space	Use artifact retention limits

Debugging Tips

1. **Enable Debug Logging:**

```yaml
- name: Run tests
  run: npx playwright test
  env:
    DEBUG: pw:api
```

2. **Upload Trace Files:**

```yaml
- name: Upload traces
  uses: actions/upload-artifact@v4
  if: failure()
  with:
    name: playwright-traces
    path: test-results/
```

3. **Check Runner Logs:**
    - Click on failed step
    - Expand to see full output
    - Look for error messages

9. Best Practices

✅ Do's

1. **Use `npm ci`** instead of `npm install` for faster, reliable installs
2. **Set reasonable timeouts** to avoid stuck workflows
3. **Upload artifacts** only when needed (use `if: failure()`)
4. **Use matrix strategy** for multi-browser testing
5. **Cache dependencies** for faster runs
6. **Use secrets** for sensitive data

❌ Don'ts

1. **Don't hardcode credentials** in workflow files
2. **Don't skip browser installation** step
3. **Don't set `fail-fast: true`** if you want all browser results
4. **Don't ignore flaky tests** - fix them!

Recommended Workflow Structure

```yaml
name: Playwright Tests

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
  workflow_dispatch:

jobs:
  test:
    timeout-minutes: 60
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - uses: actions/setup-node@v4
        with:
          node-version: 20
          cache: 'npm'  # Cache npm dependencies

      - run: npm ci

      - run: npx playwright install --with-deps

      - run: npx playwright test

      - uses: actions/upload-artifact@v4
        if: failure()  # Only on failure
        with:
          name: playwright-report
          path: playwright-report/
          retention-days: 7
```

Quick Reference

Workflow Locations

```
Repository Root
└── .github/
    └── workflows/
        └── playwright.yml  ← Your workflow file
```

Essential Commands

```
# Validate workflow syntax locally
npx yaml-lint .github/workflows/playwright.yml

# Test workflow locally (requires act)
act push

# Check git status
git status

# Push changes
git add . && git commit -m "message" && git push
```

Status Badge

Add to your README.md:

```
![Playwright Tests](https://github.com/YOUR_USERNAME/YOUR_REPO/actions/workflows/playwright.yml/
```

Summary Checklist

- [ ] Create GitHub repository
- [ ] Push local code to GitHub
- [ ] Create `.github/workflows/` directory
- [ ] Create `playwright.yml` workflow file
- [ ] Commit and push workflow

- [ ] Verify workflow runs in Actions tab
- [ ] Download and review test reports
- [ ] Add status badge to README

Document created: December 31, 2024