Browsers

Introduction

Each version of Playwright needs specific versions of browser binaries to operate. You will need to use the Playwright CLI to install these browsers.

With every release, Playwright updates the versions of the browsers it supports, so that the latest Playwright would support the latest browsers at any moment. It means that every time you update Playwright, you might need to re-run the `install` CLI command.

Install browsers

Playwright can install supported browsers. Running the command without arguments will install the default browsers.

```
npx playwright install
```

You can also install specific browsers by providing an argument:

```
npx playwright install webkit
```

See all supported browsers:

```
npx playwright install --help
```

Install system dependencies

System dependencies can get installed automatically. This is useful for CI environments.

```
npx playwright install-deps
```

You can also install the dependencies for a single browser by passing it as an argument:

```
npx playwright install-deps chromium
```

It's also possible to combine `install-deps` with `install` so that the browsers and OS dependencies are installed with a single command.

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

See system requirements for officially supported operating systems.

Update Playwright regularly

By keeping your Playwright version up to date you will be able to use new features and test your app on the latest browser versions and catch failures before the latest browser version is released to the public.

```
# Update playwright
npm install -D @playwright/test@latest

# Install new browsers
npx playwright install
```

Check the release notes to see what the latest version is and what changes have been released.

```
# See what version of Playwright you have by running the following command
npx playwright --version
```

Configure Browsers

Playwright can run tests on Chromium, WebKit and Firefox browsers as well as branded browsers such as Google Chrome and Microsoft Edge. It can also run on emulated tablet and mobile devices. See the registry of device parameters for a complete list of selected desktop, tablet and mobile devices.

Run tests on different browsers

Playwright can run your tests in multiple browsers and configurations by setting up **projects** in the config. You can also add different options for each project.

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

export default defineConfig({
  projects: [
    /* Test against desktop browsers */
    {
      name: 'chromium',
      use: { ...devices['Desktop Chrome'] },
    },
    {
      name: 'firefox',
      use: { ...devices['Desktop Firefox'] },
    },
    {
      name: 'webkit',
      use: { ...devices['Desktop Safari'] },
    },
    /* Test against mobile viewports. */
    {
      name: 'Mobile Chrome',
      use: { ...devices['Pixel 5'] },
```

```
    },
    {
      name: 'Mobile Safari',
      use: { ...devices['iPhone 12'] },
    },
    /* Test against branded browsers. */
    {
      name: 'Google Chrome',
      use: { ...devices['Desktop Chrome'], channel: 'chrome' }, // or
'chrome-beta'
    },
    {
      name: 'Microsoft Edge',
      use: { ...devices['Desktop Edge'], channel: 'msedge' }, // or
'msedge-dev'
    },
  ],
});
```

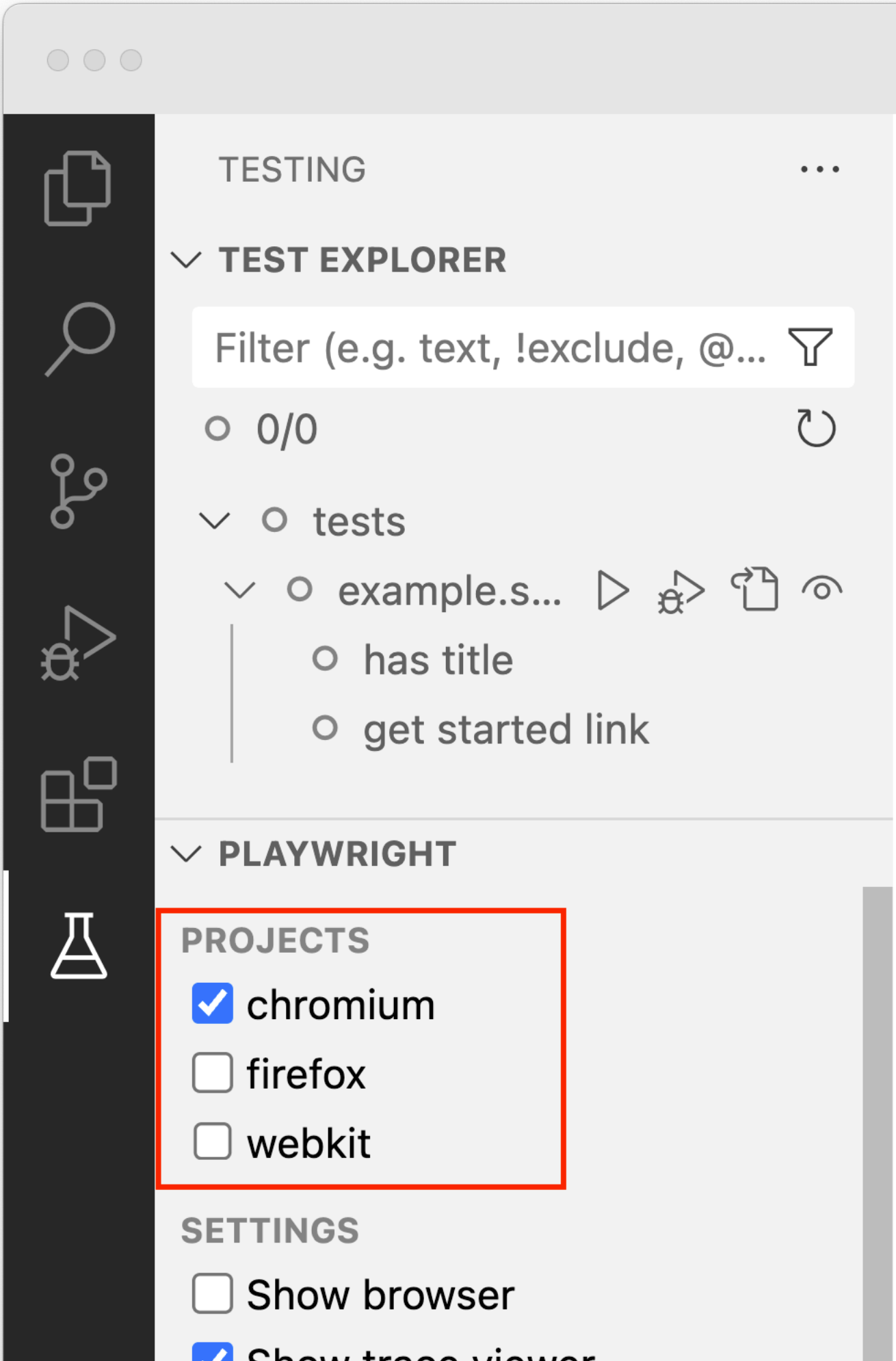
Playwright will run all projects by default.

```
npx playwright test

Running 7 tests using 5 workers

  ✓ [chromium] › example.spec.ts:3:1 › basic test (2s)
  ✓ [firefox] › example.spec.ts:3:1 › basic test (2s)
  ✓ [webkit] › example.spec.ts:3:1 › basic test (2s)
  ✓ [Mobile Chrome] › example.spec.ts:3:1 › basic test (2s)
  ✓ [Mobile Safari] › example.spec.ts:3:1 › basic test (2s)
  ✓ [Google Chrome] › example.spec.ts:3:1 › basic test (2s)
  ✓ [Microsoft Edge] › example.spec.ts:3:1 › basic test (2s)
```

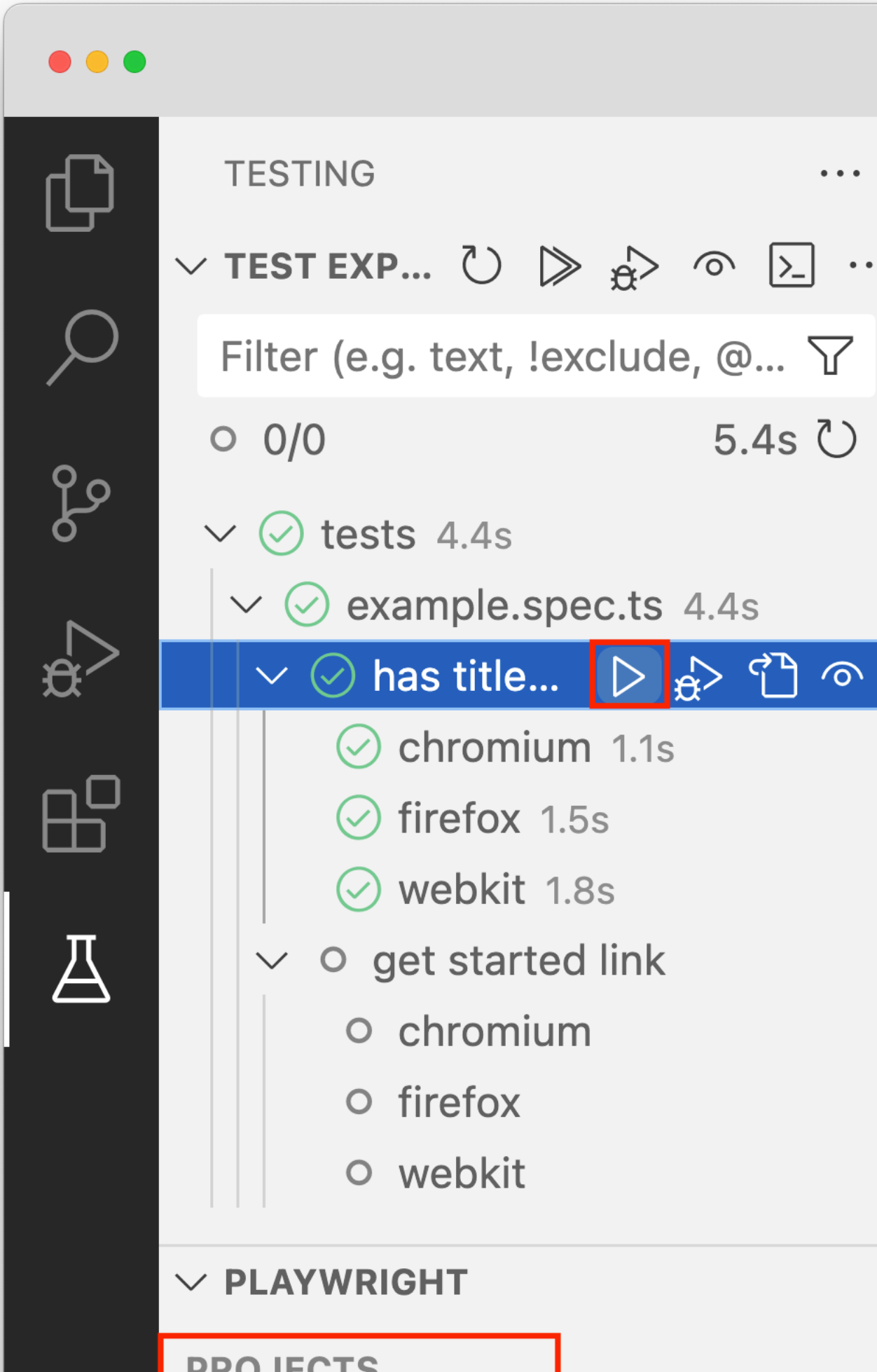
Use the `--project` command line option to run a single project.

```
npx playwright test --project=firefox

Running 1 test using 1 worker

  ✓ [firefox] › example.spec.ts:3:1 › basic test (2s)
```

With the VS Code extension you can run your tests on different browsers by checking the checkbox next to the browser name in the Playwright sidebar. These names are defined in your Playwright config file under the projects section. The default config when installing Playwright gives you 3 projects, Chromium, Firefox and WebKit. The first project is selected by default.


TESTING
TEST EXPLORER
Filter (e.g. text, !exclude, @...
0/0
tests
example.s...
has title
get started link
PLAYWRIGHT
PROJECTS
chromium
firefox
webkit
SETTINGS
Show browser

To run tests on multiple projects(browsers), select each project by checking the checkboxes next to the project name.


TESTING
TEST EXP...
Filter (e.g. text, !exclude, @...
0/0
5.4s
tests 4.4s
example.spec.ts 4.4s
has title...
chromium 1.1s
firefox 1.5s
webkit 1.8s
get started link
chromium
firefox
webkit
PLAYWRIGHT

Chromium

For Google Chrome, Microsoft Edge and other Chromium-based browsers, by default, Playwright uses open source Chromium builds. Since the Chromium project is ahead of the branded browsers, when the world is on Google Chrome N, Playwright already supports Chromium N+1 that will be released in Google Chrome and Microsoft Edge a few weeks later.

Google Chrome & Microsoft Edge

While Playwright can download and use the recent Chromium build, it can operate against the branded Google Chrome and Microsoft Edge browsers available on the machine (note that Playwright doesn't install them by default). In particular, the current Playwright version will support Stable and Beta channels of these browsers.

Available channels are `chrome`, `msedge`, `chrome-beta`, `msedge-beta` or `msedge-dev`.

WARNING

Certain Enterprise Browser Policies may impact Playwright's ability to launch and control Google Chrome and Microsoft Edge. Running in an environment with browser policies is outside of the Playwright project's scope.

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

export default defineConfig({
  projects: [
    /* Test against branded browsers. */
    {
      name: 'Google Chrome',
      use: { ...devices['Desktop Chrome'], channel: 'chrome' }, // or
'chrome-beta'
    },
    {
      name: 'Microsoft Edge',
      use: { ...devices['Desktop Edge'], channel: 'msedge' }, // or
"msedge-beta" or 'msedge-dev'
    },
  ],
});
```

Installing Google Chrome & Microsoft Edge

If Google Chrome or Microsoft Edge is not available on your machine, you can install them using the Playwright command line tool:

```
npx playwright install msedge
```

WARNING

Google Chrome or Microsoft Edge installations will be installed at the default global location of your operating system overriding your current browser installation.

Run with the `--help` option to see a full a list of browsers that can be installed.

When to use Google Chrome & Microsoft Edge and when not to?

Defaults

Using the default Playwright configuration with the latest Chromium is a good idea most of the time. Since Playwright is ahead of Stable channels for the browsers, it gives peace of mind that the upcoming Google Chrome or Microsoft Edge releases won't break your site. You catch breakage early and have a lot of time to fix it before the official Chrome update.

Regression testing

Having said that, testing policies often require regression testing to be performed against the current publicly available browsers. In this case, you can opt into one of the stable channels, `"chrome"` or `"msedge"`.

Media codecs

Another reason for testing using official binaries is to test functionality related to media codecs. Chromium does not have all the codecs that Google Chrome or Microsoft Edge are bundling due to various licensing considerations and agreements. If your site relies on this kind of codecs (which is rarely the case), you will also want to use the official channel.

Enterprise policy

Google Chrome and Microsoft Edge respect enterprise policies, which include limitations to the capabilities, network proxy, mandatory extensions that stand in the way of testing. So if you are part of the organization that uses such policies, it is easiest to use bundled Chromium for your local testing, you can still opt into stable channels on the bots that are typically free of such restrictions.

Firefox

Playwright's Firefox version matches the recent Firefox Stable build. Playwright doesn't work with the branded version of Firefox since it relies on patches.

WebKit

Playwright's WebKit is derived from the latest WebKit main branch sources, often before these updates are incorporated into Apple Safari and other WebKit-based browsers. This gives a lot of lead time to react on the potential browser update issues. Playwright doesn't work with the branded version of Safari since it relies on patches. Instead, you can test using the most recent WebKit build. Note that availability of certain features, which depend heavily on the underlying platform, may vary between operating systems.

Install behind a firewall or a proxy

By default, Playwright downloads browsers from Microsoft's CDN.

Sometimes companies maintain an internal proxy that blocks direct access to the public resources. In this case, Playwright can be configured to download browsers via a proxy server.

- Bash
- PowerShell
- Batch

```
HTTPS_PROXY=https://192.0.2.1 npx playwright install
$Env:HTTPS_PROXY="https://192.0.2.1"
npx playwright install
set HTTPS_PROXY=https://192.0.2.1
npx playwright install
```

If the requests of the proxy get intercepted with a custom untrusted certificate authority (CA) and it yields to `Error: self signed certificate in certificate chain` while downloading the browsers, you must set your custom root certificates via the `NODE_EXTRA_CA_CERTS` environment variable before installing the browsers:

- Bash
- PowerShell
- Batch

```
export NODE_EXTRA_CA_CERTS="/path/to/cert.pem"
$Env:NODE_EXTRA_CA_CERTS="C:\certs\root.crt"
set NODE_EXTRA_CA_CERTS="C:\certs\root.crt"
```

If your network is slow to connect to Playwright browser archive, you can increase the connection timeout in milliseconds with `PLAYWRIGHT_DOWNLOAD_CONNECTION_TIMEOUT` environment variable:

- Bash
- PowerShell
- Batch

```
PLAYWRIGHT_DOWNLOAD_CONNECTION_TIMEOUT=120000 npx playwright install
$Env:PLAYWRIGHT_DOWNLOAD_CONNECTION_TIMEOUT="120000"
npx playwright install
set PLAYWRIGHT_DOWNLOAD_CONNECTION_TIMEOUT=120000
npx playwright install
```

If you are installing dependencies and need to use a proxy on Linux, make sure to run the command as a root user. Otherwise, Playwright will attempt to become a root and will not pass environment variables like `HTTPS_PROXY` to the linux package manager.

```
sudo HTTPS_PROXY=https://192.0.2.1 npx playwright install-deps
```

Download from artifact repository

By default, Playwright downloads browsers from Microsoft's CDN.

Sometimes companies maintain an internal artifact repository to host browser binaries. In this case, Playwright can be configured to download from a custom location using the `PLAYWRIGHT_DOWNLOAD_HOST` env variable.

- Bash
- PowerShell
- Batch

```
PLAYWRIGHT_DOWNLOAD_HOST=http://192.0.2.1 npx playwright install
$Env:PLAYWRIGHT_DOWNLOAD_HOST="http://192.0.2.1"
npx playwright install
set PLAYWRIGHT_DOWNLOAD_HOST=http://192.0.2.1
npx playwright install
```

It is also possible to use a per-browser download hosts using `PLAYWRIGHT_CHROMIUM_DOWNLOAD_HOST`, `PLAYWRIGHT_FIREFOX_DOWNLOAD_HOST` and `PLAYWRIGHT_WEBKIT_DOWNLOAD_HOST` env variables that take precedence over `PLAYWRIGHT_DOWNLOAD_HOST`.

- Bash
- PowerShell
- Batch

```
PLAYWRIGHT_FIREFOX_DOWNLOAD_HOST=http://203.0.113.3
PLAYWRIGHT_DOWNLOAD_HOST=http://192.0.2.1 npx playwright install
$Env:PLAYWRIGHT_FIREFOX_DOWNLOAD_HOST="http://203.0.113.3"
$Env:PLAYWRIGHT_DOWNLOAD_HOST="http://192.0.2.1"
npx playwright install
set PLAYWRIGHT_FIREFOX_DOWNLOAD_HOST=http://203.0.113.3
set PLAYWRIGHT_DOWNLOAD_HOST=http://192.0.2.1
npx playwright install
```

Managing browser binaries

Playwright downloads Chromium, WebKit and Firefox browsers into the OS-specific cache folders:

- `%USERPROFILE%\AppData\Local\ms-playwright` on Windows
- `~/Library/Caches/ms-playwright` on macOS
- `~/.cache/ms-playwright` on Linux

These browsers will take a few hundred megabytes of disk space when installed:

```
du -hs ~/Library/Caches/ms-playwright/*
281M  chromium-XXXXXX
187M  firefox-XXXX
180M  webkit-XXXX
```

You can override default behavior using environment variables. When installing Playwright, ask it to download browsers into a specific location:

- Bash

- PowerShell
- Batch

```
PLAYWRIGHT_BROWSERS_PATH=$HOME/pw-browsers npx playwright install
$Env:PLAYWRIGHT_BROWSERS_PATH="$Env:USERPROFILE\pw-browsers"
npx playwright install
set PLAYWRIGHT_BROWSERS_PATH=%USERPROFILE%\pw-browsers
npx playwright install
```

When running Playwright scripts, ask it to search for browsers in a shared location.

- Bash
- PowerShell
- Batch

```
PLAYWRIGHT_BROWSERS_PATH=$HOME/pw-browsers npx playwright test
$Env:PLAYWRIGHT_BROWSERS_PATH="$Env:USERPROFILE\pw-browsers"
npx playwright test
set PLAYWRIGHT_BROWSERS_PATH=%USERPROFILE%\pw-browsers
npx playwright test
```

Playwright keeps track of packages that need those browsers and will garbage collect them as you update Playwright to the newer versions.

NOTE

Developers can opt-in in this mode via exporting `PLAYWRIGHT_BROWSERS_PATH=$HOME/pw-browsers` in their `.bashrc`.

Hermetic install

You can opt into the hermetic install and place binaries in the local folder:

- Bash
- PowerShell
- Batch

```
# Places binaries to node_modules/playwright-core/.local-browsers
PLAYWRIGHT_BROWSERS_PATH=0 npx playwright install
# Places binaries to node_modules\playwright-core\.local-browsers
$Env:PLAYWRIGHT_BROWSERS_PATH=0
npx playwright install
# Places binaries to node_modules\playwright-core\.local-browsers
set PLAYWRIGHT_BROWSERS_PATH=0
npx playwright install
```

NOTE

`PLAYWRIGHT_BROWSERS_PATH` does not change installation path for Google Chrome and Microsoft Edge.

Stale browser removal

Playwright keeps track of the clients that use its browsers. When there are no more clients that require a particular version of the browser, that version is deleted from the system. That way you can safely use Playwright instances of different versions and at the same time, you don't waste disk space for the browsers that are no longer in use.

To opt-out from the unused browser removal, you can set the `PLAYWRIGHT_SKIP_BROWSER_GC=1` environment variable.

Uninstall browsers

This will remove the browsers (chromium, firefox, webkit) of the current Playwright installation:

```
npx playwright uninstall
```

To remove browsers of other Playwright installations as well, pass `--all` flag:

```
npx playwright uninstall --all
```

Previous
Best Practices

Next
Chrome extensions

Community

- Stack Overflow
- Discord
- Twitter
- LinkedIn

More

- GitHub
- YouTube
- Blog
- Ambassadors