



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.

```
playwright install
```

You can also install specific browsers by providing an argument:

```
playwright install webkit
```

See all supported browsers:

```
playwright install --help
```

Install system dependencies

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

```
playwright install-deps
```

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

```
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.

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

See [system requirements](#) for officially supported operating systems.

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

Run tests on a specific browser:

```
pytest test_login.py --browser webkit
```

Run tests on multiple browsers:

```
pytest test_login.py --browser webkit --browser firefox
```

Test against mobile viewports:

```
pytest test_login.py --device="iPhone 13"
```

Test against branded browsers:

```
pytest test_login.py --browser-channel msedge
```

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.

```
pytest test_login.py --browser-channel msedge
```

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:

```
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 version matches the recent WebKit trunk build, before it is used in 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 against the recent WebKit build.

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**

```
pip install playwright
HTTPS_PROXY=https://192.0.2.1 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"
```

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**

```
pip install playwright  
PLAYWRIGHT_DOWNLOAD_CONNECTION_TIMEOUT=120000 playwright install
```

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**

```
pip install playwright  
PLAYWRIGHT_DOWNLOAD_HOST=http://192.0.2.1 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**

```
pip install playwright  
PLAYWRIGHT_FIREFOX_DOWNLOAD_HOST=http://203.0.113.3  
PLAYWRIGHT_DOWNLOAD_HOST=http://192.0.2.1 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**

```
pip install playwright  
PLAYWRIGHT_BROWSERS_PATH=$HOME/pw-browsers python -m 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 python playwright_script.py
```

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



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

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:

```
playwright uninstall
```

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

```
playwright uninstall --all
```