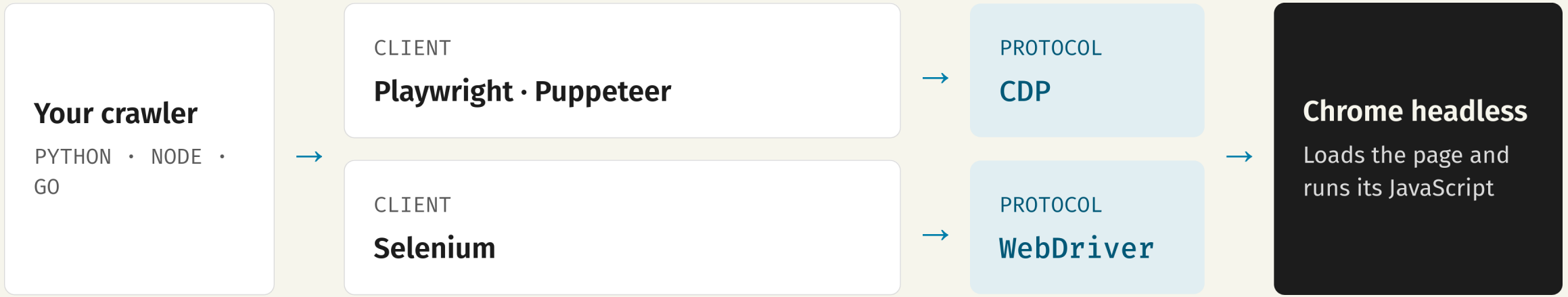
EXTRACT SUMMIT 2026

How to browse the web faster with Lightpanda



Pierre Tachoire · Lightpanda

Need JavaScript to crawl? You need a browser



Now you have a problem: you need to run Chrome in production.

01

Lightpanda

A headless browser built for machines, from scratch

Built from scratch, on proven foundations

THE ENGINE

Zig

The browser itself: DOM, event loop, CDP and WebDriver servers. Low-level language with explicit memory control.

JAVASCRIPT

V8

The same JavaScript engine as Chrome, so pages run their scripts the way they expect.

HTTP

curl

libcurl for networking: a battle-tested HTTP client used everywhere.

Not a Chromium fork. Not a WebKit patch. A new browser.

No graphical rendering, just enough composition

WHAT WE DON'T DO	WHAT WE DO: A SIMPLER COMPOSITION
Paint pixels	Compute element **size**
GPU compositing	Compute element **visibility**
Decode images, rasterise fonts	→ Faster pages: no full layout and paint pipeline
Screenshots and PDFs	→ Real interactions: clicks and visibility checks work

Crawling 933 pages: Lightpanda vs Chrome

9.7x

Faster

4.81S VS 46.70S · 25 IN PARALLEL

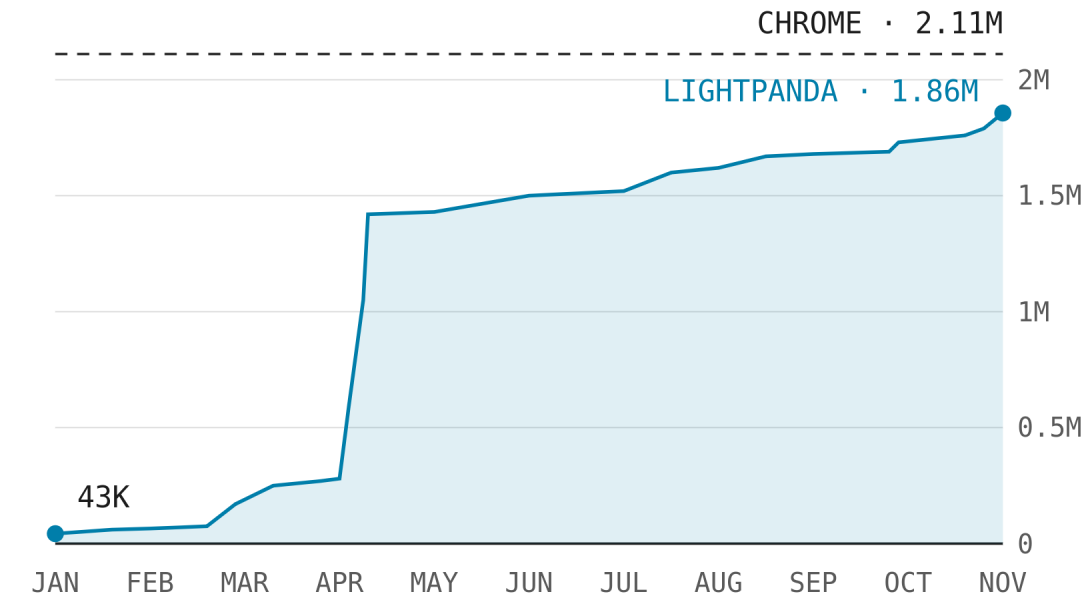
16x

Less memory

123.0M VS 2.0G PEAK · 25 IN PARALLEL

* AWS m5.xlarge · Chrome 143.0.7499.169 · Go crawler using chromedp · 933 pages of demo-browser.lightpanda.io/amiibo over the internet · 25 Lightpanda processes vs 25 Chrome tabs · github.com/lightpanda-io/demo

1.86M Web Platform subtests passing



LIGHTPANDA

85%

of all WPT subtests pass

WEB API	NOW
HTML	83%
DOM	94%
XHR	79%

Plug and play: keep your scripts

01

One binary

No dependencies to install. Or use the official Docker image.

02

Standard protocols

CDP and WebDriver: Puppeteer, Playwright, Selenium, chromedp.

03

One line changes

Point the same connect() at Lightpanda instead of Chrome. The rest stays.

START

```
$ lightpanda serve --host 127.0.0.1 --port 9222
```

PUPPETEER · LIGHTPANDA

```
const browser = await puppeteer.connect({
  browserWSEndpoint: 'ws://127.0.0.1:9222' });
```

PLAYWRIGHT · LIGHTPANDA

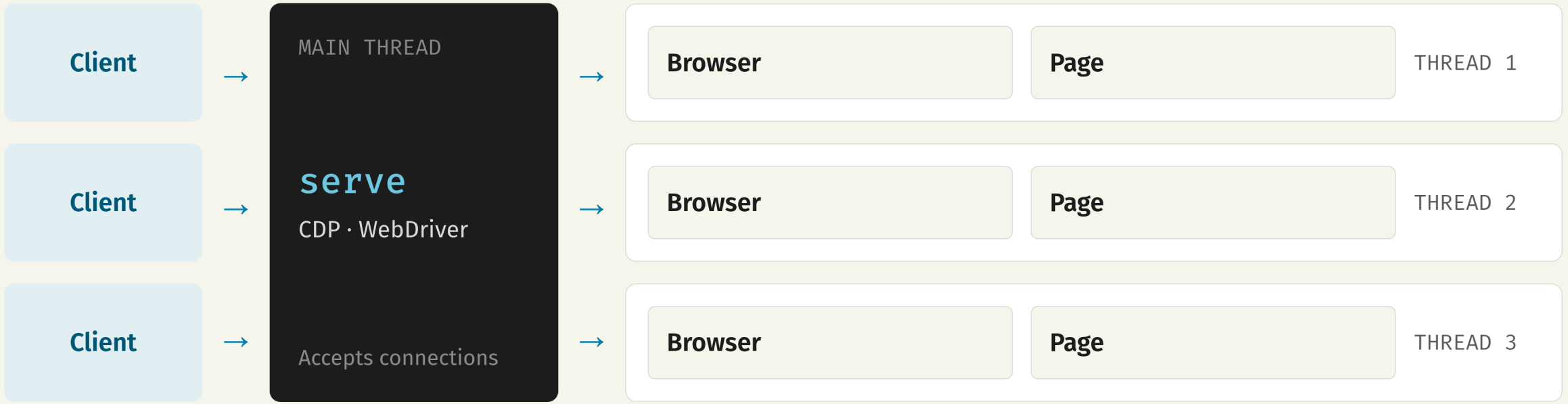
```
const browser = await chromium.connectOverCDP(
  'ws://127.0.0.1:9222');
```

02

Go faster with Lightpanda

Tune the server, fix the client, stay polite

One ephemeral browser per connection



1 connection → 1 browser

1 page at a time per browser

Ephemeral: gone when the connection closes

Pick a job lifecycle: more speed or more isolation

← FASTEST MOST ISOLATED →

01

Same connection, successive pages

Best for one website. Cookies and session state persist between pages.

02

Same connection, fresh context per job

Cookies and storage dropped with the context, without paying a reconnect.

+ Recycle the connection every N jobs to bound memory leaks

03

New connection per job

Everything discarded, memory included. Slowest: a reconnect every job.

Load only the resources you need

OFF BY DEFAULT

- Images
- Iframes
- Web Workers
- Stylesheets

OPT IN WITH `--load-resources` or `LP.configureLoading`

`iframe`

Loads iframe content. Embedded widgets, reviews and players then run their scripts and add to the page.

`worker`

Runs dedicated Web Workers. Apps that fetch or compute data off the main thread can finish rendering their content.

`stylesheet`

For computed visibility

`image`

Mimics image fetches, still no rendering

Check every page with LP.getPageStatus

```
const cdp = await page.createCDPSession();
const status = await cdp.send(
  'LP.getPageStatus');
```

```
{
  url: 'https://lightpanda.io/',
  httpStatus: 200,
  jsErrors: 0,
  unsupported: { webgl: 0, canvas2d: 0 },
  skipped: { iframe: 0, worker: 0,
    image: 33, stylesheet: 1, beacon: 0 },
  missingGlobals: [],
  failed: { script: 6, fetch: 3, xhr: 0 },
  blocked: { script: 0, fetch: 0, xhr: 0 }
}
```

CHECKS	RETURNED FIELDS
Document	`url` · `httpStatus` · `navigateError` Status code, and why the navigation failed
Bot walls	`botChallenge` · `resourceChallenge` AWS WAF, Cloudflare or Vercel, on the page or a sub-resource
Scripts	`jsErrors` · `missingGlobals` · `unsupported` Uncaught errors, undefined globals, WebGL or canvas calls
Resources	`skipped` · `failed` · `blocked` Not loaded by choice, errors ≥ 400, and the 403s and 429s

Pick the loading per context with LP.configureLoading

Turns sub-resource loading on or off for the target's browser context, from your client.

```
const cdp = await page.createCDPSession();
await cdp.send('LP.configureLoading', {
  subFrame: true,
  worker: true,
});
```

subFrame

Loads iframe documents.

worker

Starts dedicated and shared workers.

externalStylesheets

Fetches linked stylesheets. Visibility checks depend on them.

images

Fetches img sources, so load and error events reflect the real HTTP status.

More in the LP domain: fewer round-trips

EXTRACT

dump
HTML, markdown, PNG or PDF, with strip and scope

getSemanticTree
Accessibility-style tree, or a compact text outline

getStructuredData
JSON-LD, microdata, Open Graph, meta

getContentSignal
The site's ai-train, search… preferences

INSPECT

getInteractiveElements
Everything a user can act on, in order

getNodeDetails
Role, name, value and a CSS selector

detectForms
Forms and their fillable fields

ACT

clickNode
A real user click, with trusted events

fillNode
Types into inputs, textareas, selects

scrollNode
Scrolls the window or a container

handleJavaScriptDialog
Answers the next alert or prompt

WAIT

waitForSelector
Returns the node as soon as it exists

ALSO

version
Lightpanda's real version

Wait until settled, not until the network idles

`load`	**`settled`**	`networkalmostidle`	`networkidle`
Fast, may miss late content.	Waits only for work that brings content.	2 connections or fewer.	Can hang on polling.

HOLDS THE PAGE	DOESN'T HOLD
Requests started by page-load work	setInterval, timers over 100 ms
Short timers, 100 ms or less	requestAnimationFrame, IntersectionObserver
	Idle callbacks, WebSocket traffic

```
--wait-until settled · await LP.waitUntil({ until: 'settled' })
```

Usually as fast as load · at worst, networkidle

Block the ads, enable cache

`--adblock-lists easylist.txt`

EasyList syntax.

Add easylist.txt for trackers. Refresh the lists, restart to reload.

`--http-cache-dir /var/cache/lp`

One cache shared across all the browsers of the process.

A SQLite database: **one directory per process, never shared.** Capped at 1000 entries by default: tune with --http-cache-entry-limit.

Or opt in per connection with LP.configureCDP.

NEW IN 1.0.3

Compiled script cache

The HTTP cache also stores V8's compiled bytes for scripts and modules: less CPU per page.

Private cache

Caches responses that set cookies, with Set-Cookie stripped.
--experimental-features privatecache

Avoid request interception: slow and costly

Request → **Paused** → **CDP to your client** → **Your decision** → **Continue** × every request

INSTEAD

Block with ad blocker rules

```
--adblock-lists easylist.txt
--block-urls "*pattern*"
```

Blocked in the browser, before DNS.

Nothing extra to block

Lightpanda already skips images, fonts, media, iframes, workers and stylesheets.

Use the built-in cache

```
--http-cache-dir
```

Shared scripts served from disk, no client logic.

Really need it? Filter with URL patterns so only matching requests pause, and come discuss your use case with us.

The web is hostile: protect the process

Some pages loop forever or eat memory. Lightpanda's simpler composition can also take a script down a path it never takes in Chrome.

MEMORY

`--v8-max-heap-mb 200`

Maximum V8 heap per browser instance. Default: V8's own, based on available memory, so one page can take the whole process.

INFINITE LOOPS

`--watchdog-ms 2000`

Terminates JavaScript stuck without returning to the network poll. Idle or working pages never trip it. Default: a generous 30000.

Limit hit → **Browser cleared** → **Connection closed** → **Reported in /metrics and logs**

Consider WebDriver

~2x

Faster scraping over WebDriver than over CDP

SAME PAGES · LIGHTPANDA BENCHMARKS

```
$ lightpanda serve \
  --port 9222 \
  --protocol webdriver
```

WebDriver Classic or BiDi, with any WebDriver library. Same server options.

Migrating? Pass --protocol twice to serve CDP and WebDriver on one port.

Fast, but polite

GOAL	OPTION	WHY
Identify yourself	`--user-agent-suffix "MyBot/1.0"` or `--user-agent "MyBot/1.0"`	Site owners can recognise and allowlist you. No impersonation: Mozilla UAs are refused.
Respect robots.txt	`--obey-robots`	Fetches and follows each site's robots.txt.
Space out navigations	`--http-nav-delay <INT>`	Minimum ms between two top-level navigations to the same host. Default 0: off.

```
Sec-Ch-Ua: "Lightpanda";v="1"
Sec-Ch-Ua-Full-Version-List: "Lightpanda";v="1.0.2"
User-Agent: Lightpanda/1.0 MyBot/1.0
```

Client hints always say Lightpanda

Sent on every request, even when you replace the User-Agent.

Monitor with Prometheus

GET /metrics	TELLS YOU
http_duration_ms	Request latency
http_error_total	Timeouts, TLS, bot challenges
http_cache_total	Cache hit, miss, revalidated
adblock_verdicts_total	Requests the ad blocker saved
robots_access_total	robots.txt allow and deny
js_heap_size_bytes	Heap per page, to size memory
js_errors_total	Pages whose scripts break



GRAFANA · PRODUCTION FLEET · HTTP

Also exposed: CORS checks, navigation delay, WebDriver sessions, arena pools, build version.

One binary, more ways to browse

`lightpanda agent`

Drive the browser in plain English or slash commands. Save the session as a **PandaScript**.

`lightpanda.io/docs/quickstarts/agent`

`lightpanda run`

Replays a saved PandaScript, then exits. No LLM in the loop: deterministic and cheap to run in production.

`lightpanda mcp`

Starts an MCP server, over stdio or HTTP. Your LLM agents browse through tool calls.

`lightpanda fetch`

Fetches a URL, runs its JavaScript and dumps the result as HTML or markdown. One command, no client.

Give it a try and send feedback

github.com/lightpanda-io/browser

Open source (AGPL) · lightpanda.io

