04 OCTOBER 8, 2026 4 PP

Copper: the browser your agents can drive

Collin Rijock · Tim Child · Felipe Arce

We forked a small WebKit browser for the Mac and made it an MCP server, so agents work in the browser you're already signed into.

SOFTWARE-TEAM AGENTS COPPER

The problem

Our agents kept needing a browser, and they kept getting the wrong one. A fresh headless Chromium starts signed out, so the agent stalls at the first login page that matters. Driving a browser with one large-model call per click was slow, and I couldn’t see what the agent was doing until it finished or failed.

I also wanted a daily browser with Arc’s habits (spaces, a command bar, split view) that didn’t ship its own copy of Chromium.

So we built one browser for both jobs. I use it all day, and my agents drive it while I watch.

What we built

Copper is a Mac browser that Felipe Arce and I started at ShellHacks at FIU. It’s a fork of Search, an MIT-licensed WebKit browser by Office Commun. Both of its repos are public under the MIT license.

Copper runs on WebKit, the engine already inside every Mac, so it doesn’t bundle one of its own. From Arc it takes spaces with their own colors, Favorites and Saved sections, split view, tab groups and folders, a ⌘K command bar and auto-archive. One button moves you in from Arc, Chrome or Safari, and from Arc or Chrome it brings your tabs, spaces, bookmarks, history, passwords and extensions. Chrome

extensions run on WebKit's own extension engine, and Copper fills in the Chrome APIs WebKit lacks, such as bookmarks, history, downloads and the side panel.

Copper Cloud is the server half. It's one Rust daemon and one Postgres database, installed with one command on a Linux VM or with Terraform on AWS. It syncs spaces, tabs, bookmarks and history between Macs, and it hosts the live rooms behind Canvas, a shared whiteboard that opens in a tab. Sync stays off until you link an instance you run yourself, and it never carries passwords.

The biggest decision was making Copper an MCP server. Claude Code, Cursor, phi or any other MCP client can drive the browser you're already signed into. We did it because the signed-in session is the hardest part of browser automation, and the user already has one.

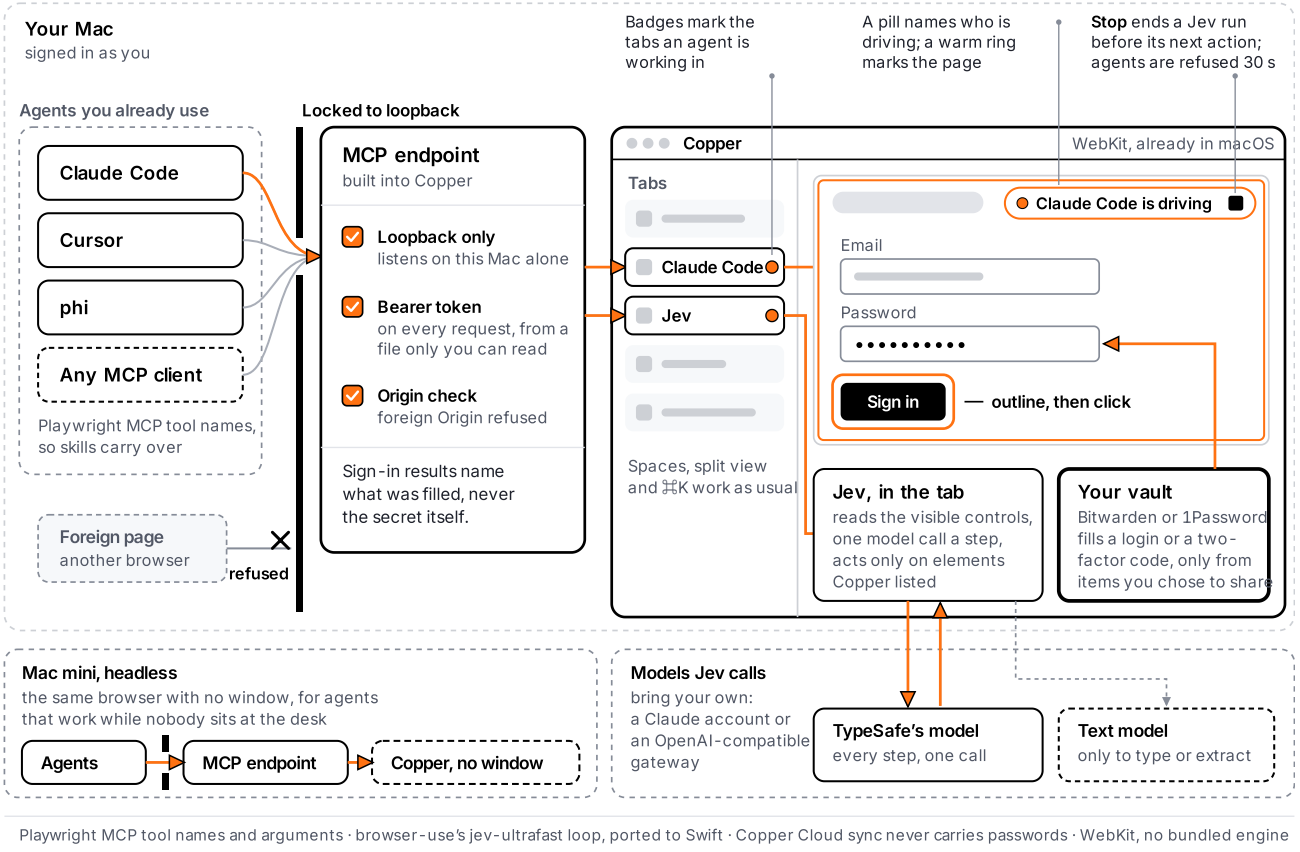


Figure 1. How agents reach the browser you're signed into. Every outside agent goes through one MCP endpoint that listens on loopback only, checks a bearer token and refuses foreign Origins. Copper shows who is driving, Jev runs inside the tab, and vault fills never hand the secret back.

How it works

The MCP endpoint listens on loopback only. Every request needs a bearer token kept in a file only the user can read, and Copper refuses any request with a foreign Origin header, so a page in another browser can’t reach in. The tool names and arguments match Playwright MCP, so a skill written for Playwright works against Copper unchanged.

For multi-step tasks Copper runs Jev. Felipe ported browser-use’s jev-ultrafast loop to Swift, and it runs inside the open tab. Each step reads the visible controls into a numbered table. One request to TypeSafe’s model then picks the operation (click, type, select, scroll, wait, done or blocked) and the element to act on. A separate text model gets called only when a step has to type something or pull data off the page. Model output never becomes a selector, a coordinate or JavaScript, so the agent can only act on elements Copper listed itself.

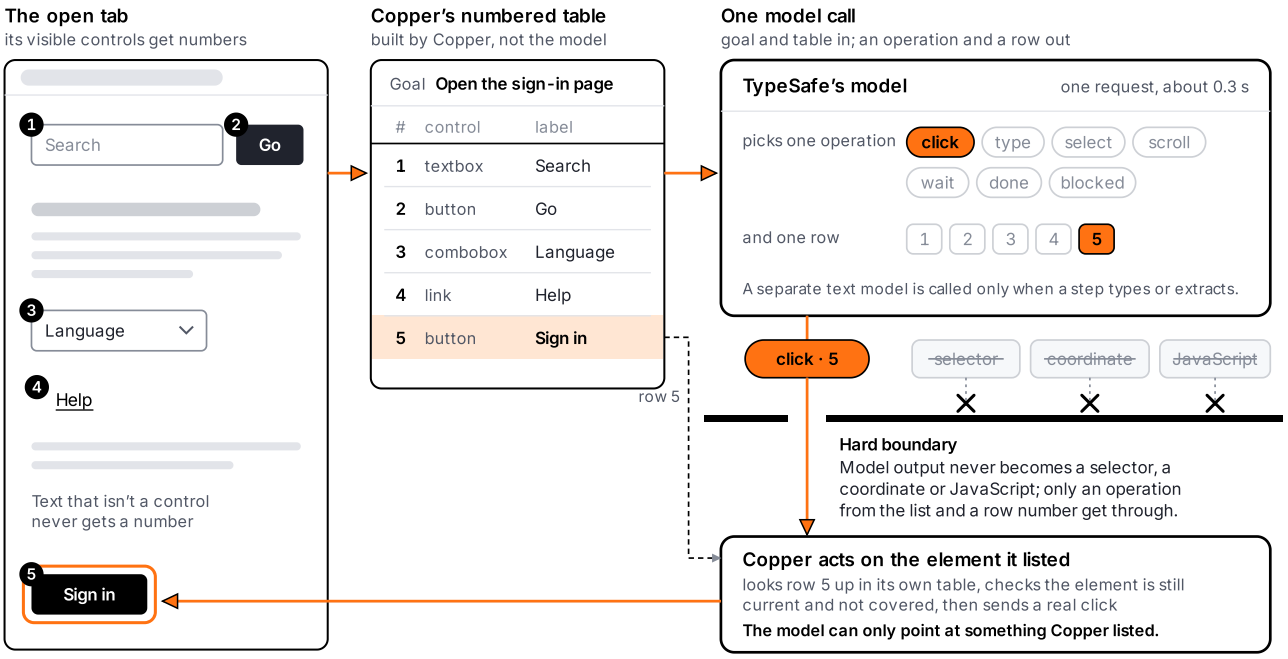


Figure 2. One Jev step on an example page. Copper numbers the visible controls, one model call answers with an operation and a row number, and Copper clicks the element it listed itself. Nothing the model writes can become a selector, a coordinate or JavaScript.

I wanted to watch agents work. Copper outlines the element an agent is about to click, puts a pill on the page naming who is driving, and badges the tabs an agent is working in. A live card in the agent pane shows each outside driver’s goal and latest

steps. Press Stop and a Jev run ends before its next action, and any other agent has its calls refused for 30 seconds.

Agents can sign in, too. Connect Bitwarden or 1Password and an agent can ask Copper to fill a saved login or a two-factor code. The tools report what was filled and never hand the secret back, and you pick which items agents may use in a setting that MCP and the CLI can't change. A headless mode runs the same browser on a Mac mini with no window, for agents that work while nobody sits at the desk.

Model access is bring-your-own: sign in with a Claude account or point Copper at an OpenAI-compatible gateway.

Results

Everything in this table comes from the two public repos, measured on October 8, 2026 (UTC). Commit counts cover the `fork` branch from the fork point on September 23, with authors grouped by name.

Measure	Value
Commits since the fork point	427
Commits by author	Tim Child 10, Felipe Arce 245, Collin Rijock 172
Releases, Oct 1 to Oct 8	29
Swift in the repo, fork point to today	19,423 to 106,767 lines
Latest download (v1.0.20261008.46)	10.6 MB zip, 20 MB installed
Copper Cloud	26,272 lines of Rust, v0.8.1

Line counts are raw `wc -l` totals, blank lines and comments included.

We benchmarked the Jev loop on September 17, before it moved into Copper. Three tasks (two on Wikipedia, one on Google Flights) ran three times each in Chrome. Jev's model picked each step in about 0.3 seconds, against about 1.4 seconds for a general-purpose model making the same decisions, and both were correct on every run. Two changes together cut the median task time from 9.3 seconds to 2.2

seconds, with all 9 runs passing: Jev chose typed text from candidates in the goal instead of calling a text model, and the loop waited for the page to settle. browser-use's own published measurement puts the median Jev decision at 178 ms.

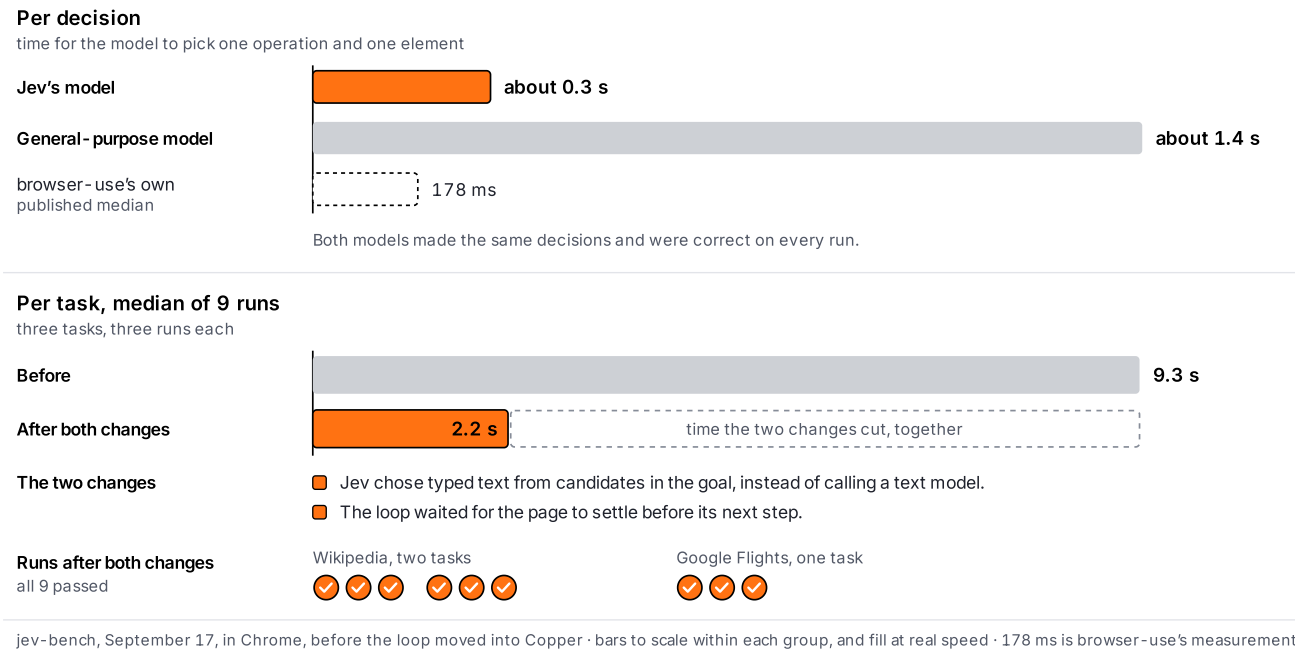


Figure 3. The Jev benchmark from September 17, three tasks run three times each in Chrome. Jev's model picks a step in about 0.3 s against about 1.4 s for a general-purpose model, and two changes cut the median task from 9.3 s to 2.2 s with all 9 runs passing. Each bar fills at the speed it measured.

What's next

The next Arc features in the fork plan are Little Arc, Peek, site search and Boosts, and none of them has started. Media controls in the sidebar sit further down the same list. Passkeys are in progress, and the plan still calls for a Liquid Glass look for macOS 26 and a widget board in the sidebar.

Both repos take issues and pull requests. The code is at github.com/copper-browser/Copper and github.com/copper-browser/copper-cloud.

About Exowatt — Exowatt is a next-generation renewable energy company providing modular dispatchable energy solutions for energy-intensive applications. Inquiries: