



# Introducing Moltworker: a self-hosted personal AI agent, minus the minis

2026-01-29



Celso Martinho



Brian Brunner



Sid Chatterjee



Andreas Jansson

9 min read

This post is also available in [\\_\\_\\_\\_\\_](#) and [한국어](#).



*Editorial note: As of January 30, 2026, Moltbot has been [renamed](#) to OpenClaw.*

The Internet woke up this week to a flood of people [buying Mac minis](#) to run [Moltbot](#) (formerly Clawdbot), an open-source, self-hosted AI agent designed to act as a personal assistant. Moltbot runs in the background on a user's own hardware, has a sizable and growing list of integrations for chat applications, AI models, and other popular tools, and can be controlled remotely. Moltbot can help you with your finances, social media, organize your day — all through your favorite messaging app.

But what if you don't want to buy new dedicated hardware? And what if you could still run your Moltbot efficiently and securely online? Meet [Moltworker](#), a middleware Worker and adapted scripts that allows running Moltbot on Cloudflare's Sandbox SDK and our Developer Platform APIs.

## A personal assistant on Cloudflare — [link](#) how does that work?

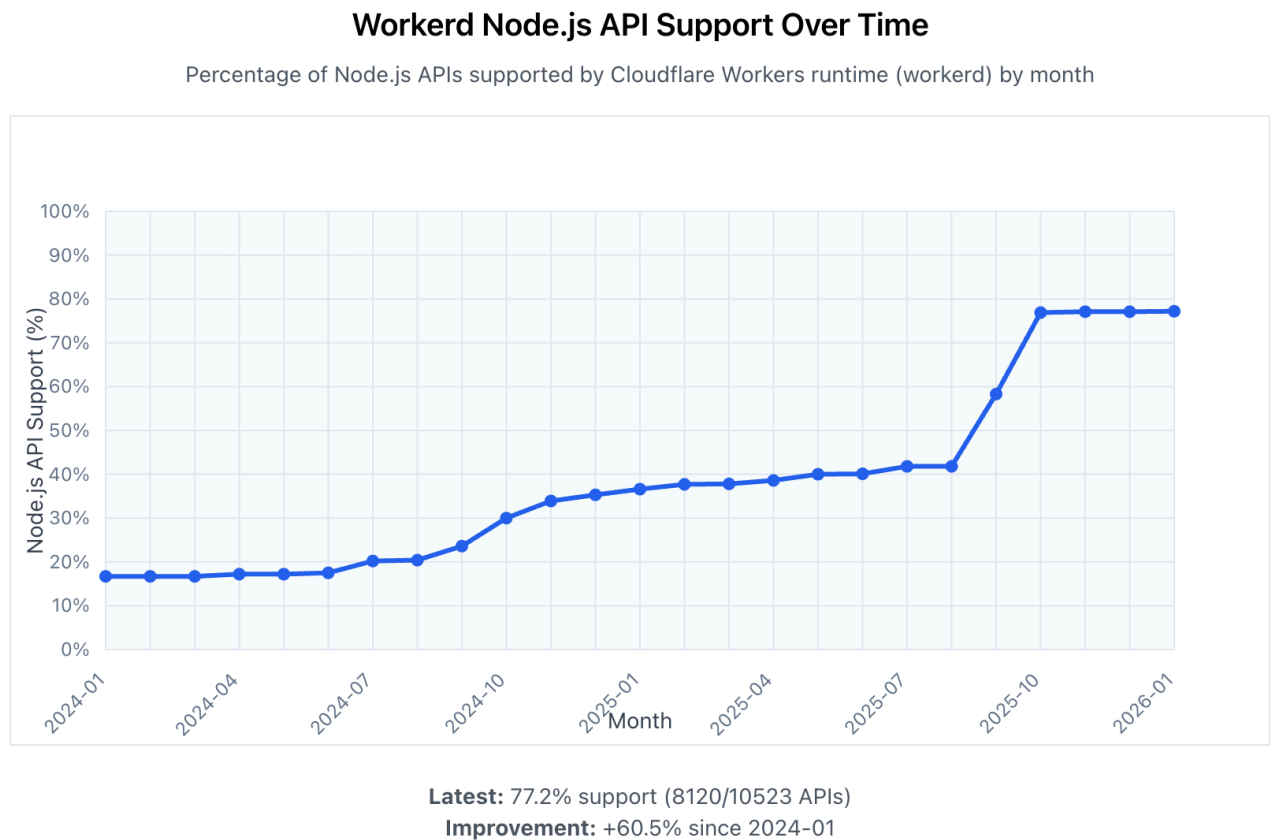
Cloudflare Workers has never been [as compatible](#) with Node.js as it is now. Where in the past we had to mock APIs to get some packages running, now those APIs are supported natively by the Workers Runtime.

This has changed how we can build tools on Cloudflare Workers. When we first implemented [Playwright](#), a popular framework for web testing and automation that runs on [Browser Rendering](#), we had to rely on [memfs](#). This was bad because not only is memfs a hack and an external dependency, but it also forced us to drift away from the official Playwright codebase. Thankfully, with more Node.js compatibility, we were able to start using [node:fs natively](#), reducing complexity and maintainability, which makes upgrades to the latest versions of Playwright easy to do.

The list of Node.js APIs we support natively keeps growing. The blog post "[A year of improving Node.js compatibility in Cloudflare Workers](#)" provides an overview of where we are and what we're doing.

We measure this progress, too. We recently ran an experiment where we took the 1,000 most popular NPM packages, installed and let AI loose, to try to run them in Cloudflare Workers, [Ralph Wiggum as a "software engineer"](#) style, and the results were surprisingly good. Excluding the packages that are build tools, CLI tools or browser-only and don't apply, only 15 packages genuinely didn't work. **That's 1.5%.**

Here's a graphic of our Node.js API support over time:



We put together a page with the results of our internal experiment on npm packages support [here](#), so you can check for yourself.

Moltbot doesn't necessarily require a lot of Workers Node.js compatibility because most of the code runs in a container anyway, but we thought it would be important to highlight how far we got supporting so many packages using native

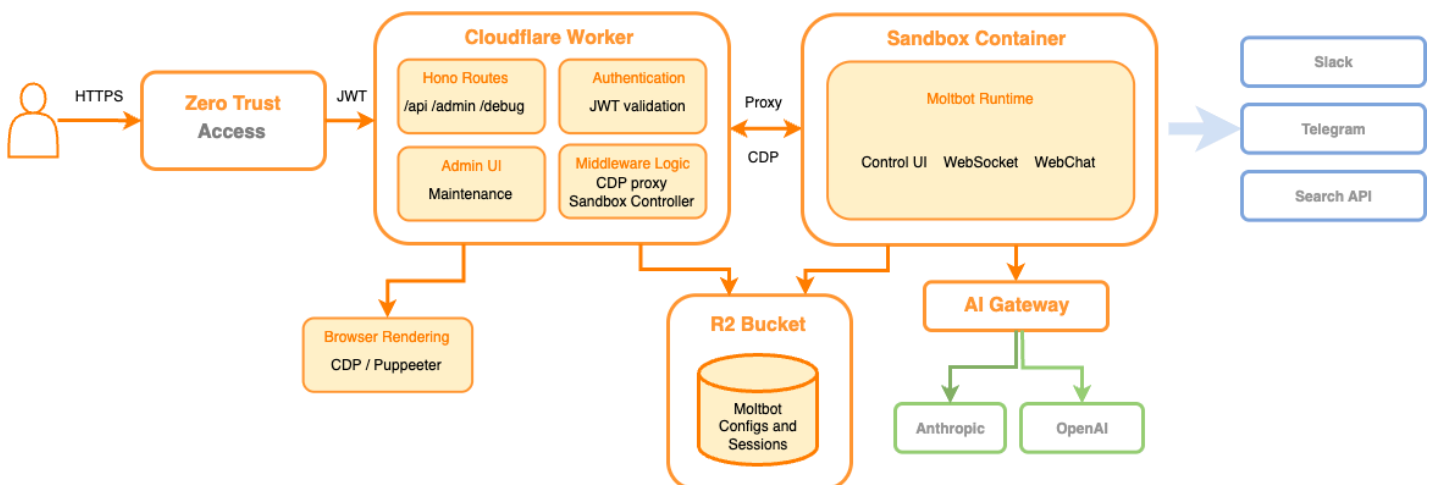
APIs. This is because when starting a new AI agent application from scratch, we can actually run a lot of the logic in Workers, closer to the user.

The other important part of the story is that the list of [products and APIs](#) on our Developer Platform has grown to the point where anyone can build and run any kind of application — even the most complex and demanding ones — on Cloudflare. And once launched, every application running on our Developer Platform immediately benefits from our secure and scalable global network.

Those products and services gave us the ingredients we needed to get started. First, we now have [Sandboxes](#), where you can run untrusted code securely in isolated environments, providing a place to run the service. Next, we now have [Browser Rendering](#), where you can programmatically control and interact with headless browser instances. And finally, [R2](#), where you can store objects persistently. With those building blocks available, we could begin work on adapting Moltbot.

## How we adapted Moltbot to run on us [🔗](#)

Moltbot on Workers, or Moltworker, is a combination of an endpoint Worker that acts as an API router and a proxy between our APIs and the isolated environment, both protected by Cloudflare Access. It also provides an administration UI and connects to the Sandbox container where the standard Moltbot Gateway runtime and its integrations are running, using R2 for persistent storage.



High-level architecture diagram of Moltworker.

Let's dive in more.

## AI Gateway [🔗](#)

Cloudflare AI Gateway acts as a proxy between your AI applications and any popular [AI provider](#), and gives our customers centralized visibility and control over the requests going through.

Recently we announced support for [Bring Your Own Key \(BYOK\)](#), where instead of passing your provider secrets in plain text with every request, we centrally manage the secrets for you and can use them with your gateway configuration.

An even better option where you don't have to manage AI providers' secrets at all end-to-end is to use [Unified Billing](#). In this case you top up your account with credits and use AI Gateway with any of the supported providers directly, Cloudflare gets charged, and we will deduct credits from your account.

To make Moltbot use AI Gateway, first we create a new gateway instance, then we enable the Anthropic provider for it, then we either add our Claude key or purchase credits to use Unified Billing, and then all we need to do is set the ANTHROPIC\_BASE\_URL environment variable so Moltbot uses the AI Gateway endpoint. That's it, no code changes necessary.

Once Moltbot starts using AI Gateway, you'll have full visibility on costs and have access to logs and analytics that will help you understand how your AI agent is using the AI providers.



Note that Anthropic is one option; Moltbot supports [other](#) AI providers and so does [AI Gateway](#). The advantage of using AI Gateway is that if a better model comes along from any provider, you don't have to swap keys in your AI Agent configuration and redeploy — you can simply switch the model in your gateway configuration. And more, you specify model or provider [fallbacks](#) to handle request failures and ensure reliability.

## Sandboxes [🔗](#)

Last year we anticipated the growing need for AI agents to run untrusted code securely in isolated environments, and we [announced](#) the [Sandbox SDK](#). This SDK is built on top of [Cloudflare Containers](#), but it provides a simple API for executing commands, managing files, running background processes, and exposing services — all from your Workers applications.

In short, instead of having to deal with the lower-level Container APIs, the Sandbox SDK gives you developer-friendly APIs for secure code execution and handles the complexity of container lifecycle, networking, file systems, and process management — letting you focus on building your application logic with just a few lines of TypeScript. Here's an example:

```
import { getSandbox } from '@cloudflare/sandbox';
export { Sandbox } from '@cloudflare/sandbox';

export default {
  async fetch(request: Request, env: Env): Promise<Response> {
    const sandbox = getSandbox(env.Sandbox, 'user-123');

    // Create a project structure
    await sandbox.mkdir('/workspace/project/src', { recursive: true });

    // Check node version
    const version = await sandbox.exec('node -v');

    // Run some python code
    const ctx = await sandbox.createCodeContext({ language: 'python' });
    await sandbox.runCode('import math; radius = 5', { context: ctx });
    const result = await sandbox.runCode('math.pi * radius ** 2', { context: ctx });
```



```
    return Response.json({ version, result });  
  }  
};
```

This fits like a glove for Moltbot. Instead of running Docker in your local Mac mini, we run Docker on Containers, use the Sandbox SDK to issue commands into the isolated environment and use callbacks to our entrypoint Worker, effectively establishing a two-way communication channel between the two systems.

## R2 for persistent storage

The good thing about running things in your local computer or VPS is you get persistent storage for free. Containers, however, are inherently [ephemeral](#), meaning data generated within them is lost upon deletion. Fear not, though — the Sandbox SDK provides the `sandbox.mountBucket()` that you can use to automatically, well, mount your R2 bucket as a filesystem partition when the container starts.

Once we have a local directory that is guaranteed to survive the container lifecycle, we can use that for Moltbot to store session memory files, conversations and other assets that are required to persist.

## Browser Rendering for browser automation

AI agents rely heavily on browsing the sometimes not-so-structured web. Moltbot utilizes dedicated Chromium instances to perform actions, navigate the web, fill out forms, take snapshots, and handle tasks that require a web browser. Sure, we can run Chromium on Sandboxes too, but what if we could simplify and use an API instead?

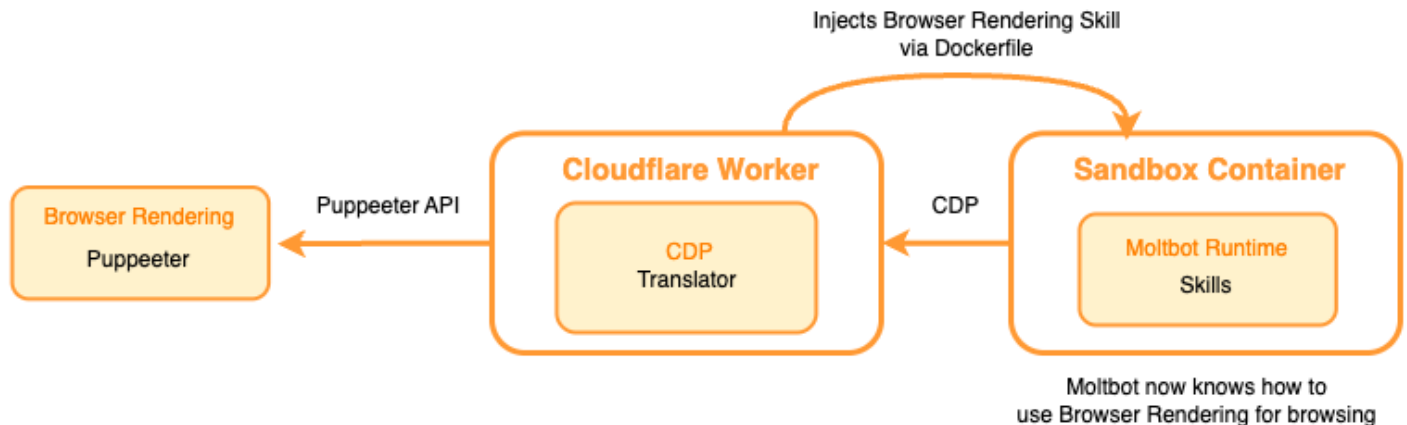
With Cloudflare's [Browser Rendering](#), you can programmatically control and interact with headless browser instances running at scale in our edge network.



We support [Puppeteer](#), [Stagehand](#), [Playwright](#) and other popular packages so that developers can onboard with minimal code changes. We even support [MCP](#) for AI.

In order to get Browser Rendering to work with Moltbot we do two things:

- First we create a [thin CDP proxy](#) (CDP is the protocol that allows instrumenting Chromium-based browsers) from the Sandbox container to the Moltbot Worker, back to Browser Rendering using the Puppeteer APIs.
- Then we inject a [Browser Rendering skill](#) into the runtime when the Sandbox starts.



From the Moltbot runtime perspective, it has a local CDP port it can connect to and perform browser tasks.

## Zero Trust Access for authentication policies [🔗](#)

Next up we want to protect our APIs and Admin UI from unauthorized access. Doing authentication from scratch is hard, and is typically the kind of wheel you don't want to reinvent or have to deal with. Zero Trust Access makes it incredibly easy to protect your application by defining specific policies and login methods for the endpoints.

[← Back to Applications](#)

# Edit moltworker

Basic information Policies **Login methods** Experience settings Advanced settings

## Login methods

Select the identity providers your users can use to log in to this application. If you do not add an identity provider, a one time pin will be used by default.

[Manage login methods.](#) 

### Accept all available identity providers



Allow users to log in with any identity providers configured on your account, including ones added in the future.

Showing 1 - 1

 Search



Name



 One-time PIN  
\*\*\*\*

1 - 1 | Items per page: 20

< 1 of 1 page >

### Instant Auth

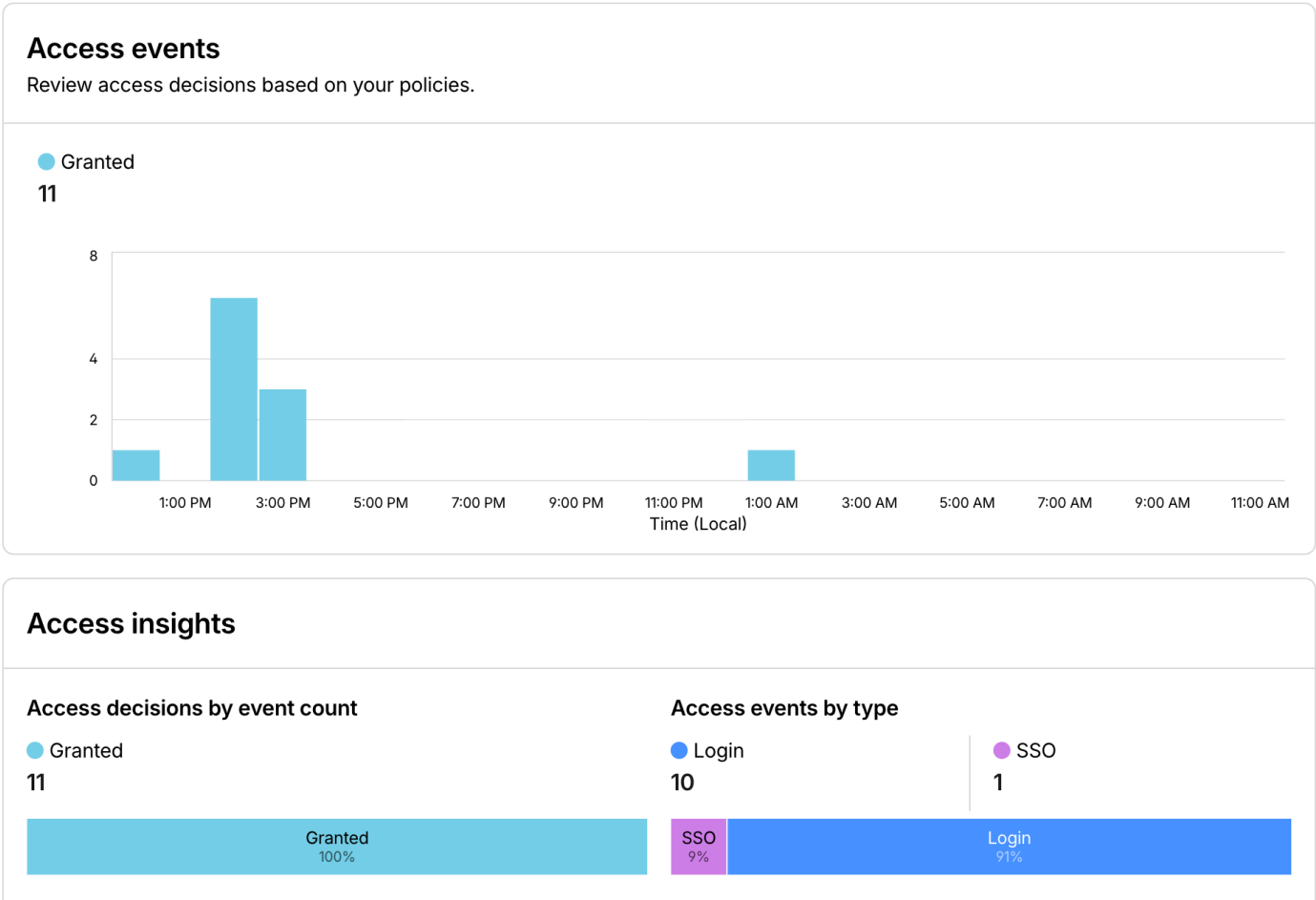


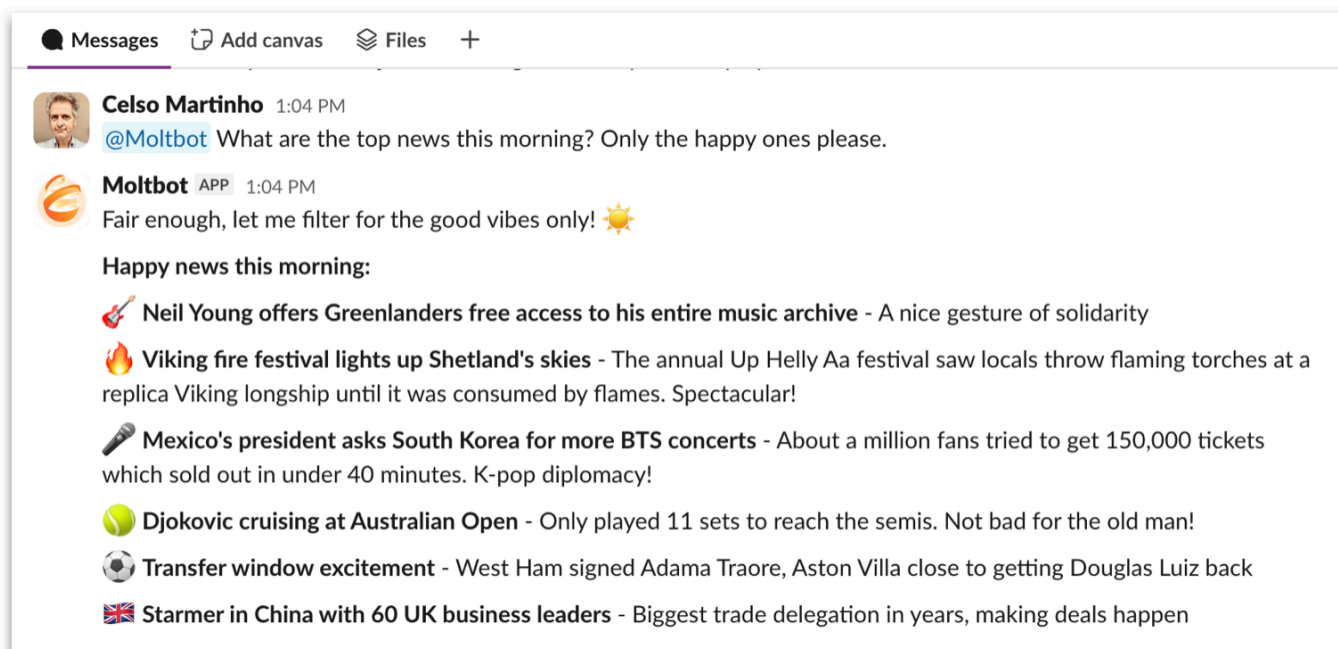
Allow users to skip identity provider selection when only one login method is available.

Zero Trust Access Login methods configuration for the Moltworker application.

Once the endpoints are protected, Cloudflare will handle authentication for you and automatically include a [JWT token](#) with every request to your origin endpoints. You can then [validate](#) that JWT for extra protection, to ensure that the request came from Access and not a malicious third party.

Like with AI Gateway, once all your APIs are behind Access you get great observability on who the users are and what they are doing with your Moltbot instance.





Here's a chat session where we ask Moltbot to find the shortest route between Cloudflare in London and Cloudflare in Lisbon using Google Maps and take a screenshot in a Slack channel. It goes through a sequence of steps using Browser Rendering to navigate Google Maps and does a pretty good job at it. Also look at Moltbot's memory in action when we ask him the second time.

We're in the mood for some Asian food today, let's get Moltbot to work for help.



**Celso Martinho** 1:14 PM

@Moltbot I'm in Lisbon today, can you search the web for best asian food restaurant in town?



**Moltbot** APP 1:14 PM

Let me try a different approach:

I found some useful info. Let me search for more specific Asian restaurants:

Based on what I found in the search, let me compile the best recommendations:

Sorry, having trouble with some search APIs today! But based on what I could gather from TripAdvisor and local guides, here are some top Asian restaurants in Lisbon:

 **Japanese/Nikkei:**

- **Ponja Nikkei Chiado** - Japanese-Peruvian fusion (\$\$\$\$ , upscale)
- **Go Juu** - Great sushi and Japanese cuisine in Príncipe Real

 **Indian/Nepalese:**

- **Ola Nepal House** - Very popular, currently 30% off on TripAdvisor
- **Sanskar Nepal** - Budget-friendly Nepalese in Mouraria (\$ range)
- **Ola Nepal Marquês de Pombal** - Another branch, also discounted

We eat with our eyes too.

Let's get more creative and ask Moltbot to create a video where it browses our developer documentation. As you can see, it downloads and runs ffmpeg to generate the video out of the frames it captured in the browser.

## Run your own Moltworker<sup>🔗</sup>

We open-sourced our implementation and made it available at <https://github.com/cloudflare/moltworker>, so you can deploy and run your own Moltbot on top of Workers today.

The [README](#) guides you through the necessary steps to set up everything. You will need a Cloudflare account and a minimum \$5 USD [Workers paid plan](#) subscription to use Sandbox Containers, but all the other products are either free to use, like [AI Gateway](#), or have generous [free tiers](#) you can use to get you started and run for as long as you want under reasonable limits.

**Note that Moltworker is a proof of concept, not a Cloudflare product.** Our goal is to showcase some of the most exciting features of our [Developer Platform](#) that can be used to run AI agents and unsupervised code efficiently and securely, and get great observability while taking advantage of our global network.



Feel free to contribute to or fork our [GitHub](#) repository; we will keep an eye on it for a while for support. We are also considering contributing upstream to the official project with Cloudflare skills in parallel.

## Conclusion [↗](#)

We hope you enjoyed this experiment, and we were able to convince you that Cloudflare is the perfect place to run your AI applications and agents. We've been working relentlessly trying to anticipate the future and release features like the [Agents SDK](#) that you can use to build your first agent [in minutes](#), [Sandboxes](#) where you can run arbitrary code in an isolated environment without the complications of the lifecycle of a container, and [AI Search](#), Cloudflare's managed vector-based search service, to name a few.

Cloudflare now offers a complete toolkit for AI development: inference, storage APIs, databases, durable execution for stateful workflows, and built-in AI capabilities. Together, these building blocks make it possible to build and run even the most demanding AI applications on our global edge network.

If you're excited about AI and want to help us build the next generation of products and APIs, we're [hiring](#).

---

Cloudflare's connectivity cloud protects [entire corporate networks](#), helps customers build [Internet-scale applications efficiently](#), accelerates any [website or Internet application](#), [wards off DDoS attacks](#), keeps [hackers at bay](#), and can help you on [your journey to Zero Trust](#).

Visit [1.1.1.1](#) from any device to get started with our free app that makes your Internet faster and safer.

To learn more about our mission to help build a better Internet, [start here](#). If you're looking for a new career direction, check out [our open positions](#).

[Discuss on Hacker News](#)

[AI](#) [Agents](#) [Cloudflare Workers](#) [Containers](#) [Sandbox](#)

---

## Follow on X

Celso Martinho | [@celso](#)

Sid Chatterjee | [@chatsidhartha](#)

Cloudflare | [@cloudflare](#)

---

## RELATED POSTS

February 05, 2026 9:00 AM

### 2025 Q4 DDoS threat report: A record-setting 31.4 Tbps attack caps a year of massive DDoS assaults

The number of DDoS attacks more than doubled in 2025. The network layer is under particular threat as hyper-volumetric attacks grew 700%....

**By Omer Yoachimik, Jorge Pacheco, Cloudforce One**

[DDoS Reports](#), [DDoS](#), [Cloudforce One](#), [Security](#), [Advanced DDoS](#), [AI](#)

January 30, 2026 12:01 PM

## Google's AI advantage: why crawler separation is the only path to a fair Internet

Google's dual-purpose crawler creates an unfair AI advantage. To protect publishers and foster competition, the UK's Competition and Markets Authority must mandate crawler separation for search and AI....

**By Maria Palmieri, Sebastian Hufnagel**

[AI](#), [AI Bots](#), [Google](#), [Legal](#), [Policy & Legal](#)

January 30, 2026 9:00 AM

## Building vertical microfrontends on Cloudflare's platform

Deploy multiple Workers under a single domain with the ability to make them feel like single-page applications. We take a look at how service bindings enable URL path routing to multiple projects....

**By Brayden Wilmoth**

[Cloudflare Workers](#), [Developer Platform](#), [Developers](#), [Dashboard](#), [Front End](#), [Micro-frontends](#)

January 27, 2026 9:00 AM

## Building a serverless, post-quantum Matrix homeserver

As a proof of concept, we built a Matrix homeserver to Cloudflare Workers — delivering encrypted messaging at the edge with automatic post-quantum cryptography....

**By Nick Kuntz**

[Cloudflare Workers](#), [Durable Objects](#), [D1](#), [Cloudflare Workers KV](#), [R2](#), [Security](#), [Developer Platform](#), [Developers](#), [Rust](#), [WebAssembly](#), [Post-Quantum](#), [Encryption](#)

