

The Free Infrastructure Layer

Host every website, every file, every customer portal, every scheduled task, and every AI agent in your business on Cloudflare. Same stack Disney and Shopify run on, free at the scale most of us are at. Replaces Squarespace, Mailchimp, Dropbox, Zapier, and Jobber portals with one plug-in.

Built by [@brycenwood.ai](#) · October 2026

THE PROBLEM

You are paying for stuff this plug-in already does.

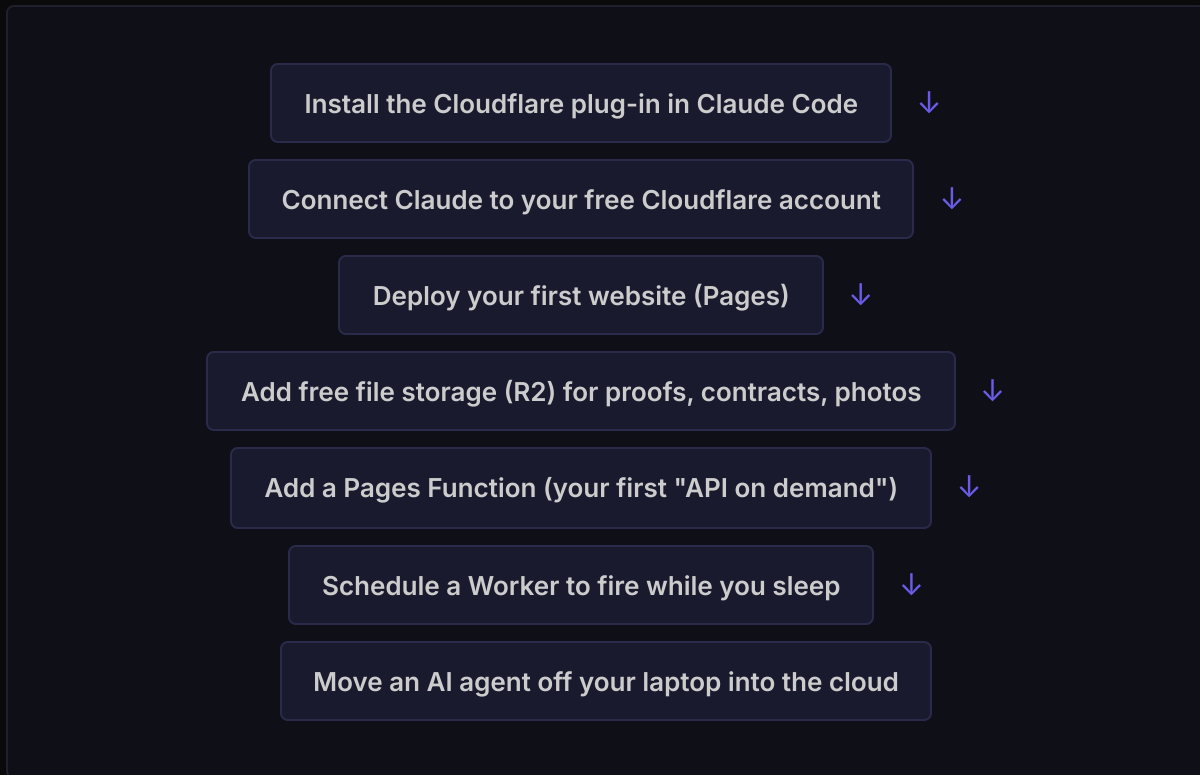
Walk through the SaaS stack of any service business and you will see the same names every time. Squarespace or Wix for the website (\$20-30/mo). Mailchimp or Klaviyo for transactional email (\$20-50/mo). Dropbox or Google Drive for file storage (\$10-20/mo). Zapier for automations (\$30-50/mo). Jobber's portal add-on for customer login (\$69+/mo). Lighthouse or SpeedCurve for site speed tracking (\$50+/mo).

That is \$200-500 a month in tools. Most of them duplicating things a single piece of infrastructure already does — for free, at the volume most of us operate at.

System 9 is that infrastructure. Cloudflare. The same network that runs Shopify checkouts, Disney+, and most of the modern internet. They give it away at small scale because most owners never even discover it exists. Inside Claude Code, there is a plug-in that connects directly to it. Once it is installed, every other system in this playbook gets a free home.

What You're Building

A single infrastructure layer that hosts all of the systems you already built in this playbook. Your llms.txt site (Systems 1-3). Your full website (System 5). Your CRM lead recovery cron (System 6). Your AI executive team (System 8). Anywhere those needed a "where does this run" answer, this is the answer.



HOW THIS STACKS WITH THE REST OF THE PLAYBOOK

Systems 1-8 are the "what" of your business operating system. System 9 is the "where it runs." Your llms.txt site (System 1) lives on Pages. Your full website (System 5) lives on Pages. Your morning lead-recovery cron (System 6) can run as a scheduled Worker. Your AI executive team (System 8) can move off your laptop and into Cloudflare Workers when you want them firing while you sleep. Build System 9 last, then bring every other system home to it.

What You Need

- **Claude Code** installed (the CLI tool, not claude.ai)

- **A free Cloudflare account.** Sign up at cloudflare.com. No credit card needed for the free tier.
- **A GitHub account** (also free). Cloudflare Pages auto-deploys from GitHub repos, so this is how your sites go live.
- **The Cloudflare plug-in** enabled in Claude Code (we install this in Step 1).
- **One existing website or project** you want to host. If you do not have one yet, build System 5 first — it generates a 40-page site you can drop straight onto Cloudflare.

TIME + COST

End to end: about 90 minutes to install the plug-in, connect Cloudflare, deploy your first site, and add storage. After that, every new site or function is 5-15 minutes.

Cost at small business scale: \$0/month. The free tier covers 100,000 requests per day, unlimited static sites, 10GB of file storage, and 100,000 Worker requests. Most service businesses never touch the paid tier.

STEP 1

Install the Cloudflare plug-in

Open Claude Code in your terminal. Inside Claude Code there is a plug-in marketplace most users never open. The Cloudflare plug-in adds dozens of skills and direct API access to your Cloudflare account, so Claude can deploy, configure, and operate everything for you in plain English.

PASTE THIS INTO CLAUDE CODE

I want to install the Cloudflare plug-in inside Claude Code so I can deploy and operate websites, file storage, and AI agents on Cloudflare's free tier.

Step 1: Open the plug-in marketplace. The command is `/plugin` in Claude Code.

Step 2: Find the plug-in named "cloudflare" published by `claude-plugins-official`. Install it.

Step 3: After it installs, type `/cloudflare` and confirm the autocomplete shows skills like `/wrangler`, `/agents-sdk`, `/durable-objects`, `/cloudflare-email-service`, `/web-perf`, `/workers-best-`

practices

Step 4: Confirm it is enabled by running `/plugin` again and searching "cloudflare" – you should see the plug-in plus 5 connected MCPs (cloudflare-api, cloudflare-docs, cloudflare-bindings, cloudflare-builds, cloudflare-observability) all marked enabled or connected.

Confirm the install with me before moving on.

WHY THE PLUG-IN MATTERS

You can use Cloudflare without the plug-in. But every time you do anything, you would have to give Claude a tutorial. The plug-in pre-loads expert-level knowledge of every part of Cloudflare's platform — Workers, Pages, R2, Durable Objects, Email Service — so Claude can build, deploy, and debug in one prompt instead of ten. It is the difference between hiring a developer who already knows your stack vs one who has to be onboarded every conversation.

STEP 2

Connect Claude to your Cloudflare account

The plug-in is installed but it does not yet know which Cloudflare account it is operating on. We connect that next.

PASTE THIS INTO CLAUDE CODE

I have the Cloudflare plug-in installed. Now I need to authenticate it against my Cloudflare account.

Walk me through authenticating the cloudflare-api MCP. Tell me what to click, what to authorize, and how to confirm the connection is live.

After we authenticate, run a sanity check: list every account I have access to inside Cloudflare. Confirm the account ID matches my account.

Then list any Pages projects, R2 buckets, and Workers I already have. If this is a brand new account, the list will be empty – that is fine.

ONE ACCOUNT, MANY SITES

You only need one Cloudflare account. From that single account you can host unlimited static sites, unlimited subdomains, multiple R2 storage buckets, and multiple Workers. Most service business owners I work with run their entire business stack — main website, customer portal, internal admin tool, scheduled jobs — out of one Cloudflare account.

STEP 3

Deploy your first website

Your first site on Cloudflare Pages. If you built System 5 (the full website builder), you already have a 40-page site sitting in a folder. We are going to put that site online, for free, in under 10 minutes.

PASTE THIS INTO CLAUDE CODE

I want to deploy my website to Cloudflare Pages, which is their free static site hosting.

The site lives at [PATH TO YOUR BUILD FOLDER — e.g. ~/Documents/MyBusiness/website/build].

Do the following:

1. Create a new GitHub repo for this site. Name it [my-business-name]-website.
2. Initialize git in the build folder, commit the current state, and push to the new repo.
3. In my Cloudflare account, create a new Pages project that connects to that GitHub repo. Build command: none (it is already static HTML). Output directory: / (root).
4. Trigger the first deploy.
5. Show me the live URL when it is up.

After it is live, walk me through pointing my custom domain (e.g. mybusiness.com) at the Pages project. Show me the exact DNS records I need to add.

GIT-PUSH DEPLOYS

Cloudflare Pages auto-deploys every time you push to GitHub. That means once it is wired up, you never "deploy" again. You just commit and push. Claude can do both of those for you in one prompt. The site updates within 30 seconds of the

push. Total time from "I want to change this hero headline" to "it is live on the internet" is about 60 seconds.

STEP 4

Add free file storage (R2)

Every service business needs somewhere to put files — design proofs, signed contracts, customer photos, invoice PDFs, before-and-after shots. Dropbox charges for it. Cloudflare's storage product, R2, gives you 10GB free and charges nothing for downloads. We use it to store proofs my designer uploads from his phone, and it serves them straight to customers without any middleware.

PASTE THIS INTO CLAUDE CODE

I want to add file storage to my business so I can store customer photos, proofs, contracts, and other files for free. We are going to use Cloudflare R2 since it has 10GB free and no egress fees.

Step 1: Create a new R2 bucket in my Cloudflare account. Name it [my-business-files] or similar.

Step 2: Generate an R2 API token I can use to upload files programmatically. Save the credentials to my project's .env file (and add .env to .gitignore so it is not committed).

Step 3: Write me a small Python script that uploads a single file from my computer to the R2 bucket and prints the public URL when done. Show me the script.

Step 4: Test the upload with a sample file. Confirm the file is in the bucket and the URL works.

PUBLIC VS PRIVATE FILES

By default R2 buckets are private — only your code can read from them. If you want a file to be publicly accessible (a design proof you are sending to a customer to approve, for example), Claude can wire up a Pages Function that fetches the file from R2 and serves it with a clean URL like `yoursite.com/proofs/2026-05-16/job-1234/proof-1.png`. We cover Pages Functions in the next step.

Add a Pages Function (your first "API on demand")

This is the moment Cloudflare stops being a "static host" and starts being your business infrastructure. Pages Functions let you add server-side logic to your site — form handlers, payment processors, file uploaders, AI calls — without standing up a separate backend. You write a JavaScript file in a special `functions/` folder inside your repo. Cloudflare wires it up automatically. It runs every time someone hits the matching URL.

PASTE THIS INTO CLAUDE CODE

I want to add a Pages Function to my website that handles contact form submissions. When a customer fills out my contact form, the function will:

1. Receive the POST request from the form
2. Validate the fields (name, email, message)
3. Send me an email with the lead details using Cloudflare's free Email Service
4. Push the lead into my GHL CRM via the API (if I have GHL set up from System 7)
5. Return a friendly success message to the customer

Build the function at `functions/api/contact.js` inside the website repo. Configure my contact form to POST to `/api/contact`. Commit and push. Verify it deploys.

Test it by submitting the form once. Show me the email I should receive and confirm the lead landed in GHL.

WHY THIS IS THE UNLOCK

Most service business owners think they need a backend developer to add anything dynamic to their site. They don't. A Pages Function is a 50-line file that handles one job. You can have 100 of them. Each one is free up to 100,000 requests per day. My business uses them for contact forms, payment-related webhooks, design proof uploads, scheduled-task triggers, and dozens of small "this needs to happen when X" jobs. Claude writes them in plain English.

Schedule a Worker to fire while you sleep

A scheduled Worker is a JavaScript file that runs on a cron schedule, inside Cloudflare's network, 24/7. It needs no laptop. It needs no server. You write the logic, set the schedule, and it fires forever. This is how System 6 (CRM Lead Recovery) graduates from "runs on my laptop when I am awake" to "runs at 5am every morning whether I am there or not."

PASTE THIS INTO CLAUDE CODE

I want to move the System 6 morning lead-recovery cron from my laptop into Cloudflare as a scheduled Worker. It will run every morning at 5am, regardless of whether my laptop is on.

The Worker should:

1. Authenticate against my GHL account using the credentials in my Cloudflare secrets (we add these in this step)
2. Pull every contact added in the last 48 hours that has no opportunity attached
3. For each one, generate a personalized follow-up email draft using the Anthropic API
4. Either push the drafts into my Gmail "drafts" folder, or post a summary message to my Slack – your call which is cleaner
5. Log every run + result to a small KV store so I can audit it later

Build the Worker. Set up Cloudflare secrets for the GHL API key + Anthropic API key. Wire the cron trigger to fire at 5am Mountain Time daily. Deploy. Show me the URL where I can view the Worker's logs.

Test it once on demand to confirm it works before relying on the schedule.

THIS IS THE AUTOPILOT

You stop being the bottleneck the moment your operational scripts run somewhere that is not your laptop. A scheduled Worker fires whether you are sleeping, in the gym, on a date, or on a flight. The first one is the hardest. After you have one, you will keep adding them. Reactivation campaigns, weekly reports, customer-success check-ins, abandoned-cart recovery, daily backups. Each new one is 10-15 minutes with Claude.

Move an AI agent into the cloud

This is the boss-level move. Your AI executive team from System 8 currently runs on your laptop, inside your Claude Code session. Cloudflare's Agents SDK lets you take one of those agents and put it in the cloud, where it runs 24/7, watches webhook events from your CRM or website, and responds to them in real time. The customer sends a message at 2am? The agent reads it, drafts a reply, sends it. You wake up to a one-line summary.

PASTE THIS INTO CLAUDE CODE

```
I want to move one of my agents from System 8 – specifically my "closer agent" that drafts hot inbound replies – out of my laptop and into Cloudflare so it can respond to GHL inbound messages in real time, 24/7.
```

```
Use Cloudflare's Agents SDK. Build a Cloudflare Worker that:
```

1. Receives a webhook from GHL every time a new SMS or DM comes in
2. Reads the conversation history for that contact
3. Classifies the message: cold tire-kicker, hot inquiry, support, or spam
4. For hot inquiries: drafts a personalized reply using the same closer-agent charter from System 8 and posts the draft to my Slack #inbound channel for one-tap approval
5. For everything else: tags the contact appropriately in GHL and notifies me daily, not instantly

```
Build it. Deploy it. Set the webhook URL inside GHL to point at the new Worker. Test with a real inbound message.
```

```
This Worker is allowed to read GHL and post to Slack, but is NOT allowed to send any messages directly to the customer. Drafts only. I approve the send.
```

EARN THE AUTO-PILOT

Even when an agent is in the cloud, default it to draft-only for the first 30-60 days. You want to watch what it would have done before you let it send. Once you have 30 days of solid drafts you would have approved anyway, you can flip a switch to let it auto-send for the narrow message types you trust. Most owners get to 50-70% auto-send within 90 days. The other 30-50% stays in human-in-the-loop forever, by design.

Three more dispatches to run today

Once Cloudflare is wired in, you will keep finding things to move to it. Three to start with.

PASTE THIS INTO CLAUDE CODE

Run a Cloudflare web-perf audit on my main website. Measure Core Web Vitals (LCP, INP, CLS), find any render-blocking resources, and list the top 3 fixes ranked by impact. Show me the report. Then ask me which fix to implement first.

PASTE THIS INTO CLAUDE CODE

Build me a Pages Function that powers a "Get a Quote" form on my website. When a customer submits the form, the function should email me the lead, push it into GHL with the tag "quote-request", send the customer an auto-reply with my typical response time, and log the request to a KV store I can audit weekly. Walk me through every line. Test with a sample submission.

PASTE THIS INTO CLAUDE CODE

Build me a scheduled Worker that runs every Monday at 7am Mountain Time, pulls my last 7 days of website traffic from Cloudflare Analytics, pulls my last 7 days of leads from GHL, and posts a one-message weekly summary to my Slack: "{N} visitors, {N} leads, {N} closed, top traffic source: {X}, top converting page: {Y}." Build it, deploy it, fire it once on demand to confirm.

REAL RESULT

I run 4 sites on Cloudflare. My personal brand (brycenwood.com). My vehicle wrap company (summitwrapsandgraphics.com). My meal prep company (gainztrainprep.com). My designer's customer portal (designer.summitwrapsandgraphics.com). All on the free tier. Total monthly bill: \$0. They host the websites, they store every design proof my installer uploads from his phone, they handle every contact form, they fire scheduled jobs every morning,

and they power three of my AI agents that respond to customer messages in real time. Before Cloudflare I was paying Squarespace + Mailchimp + Dropbox + Zapier roughly \$180/month. Now it is one plug-in inside Claude Code. Built every piece of this from a couch with zero coding experience.

That is the whole system.

Zero ad spend. Zero coding experience. The other playbooks
live at brycenwood.com/playbook, one download each.

Built by Brycen Wood · [@brycenwood.ai](https://twitter.com/brycenwood.ai) · brycenwood.com