THE FREE PLAYBOOK

The AI SEO Playbook

Ten systems that put my business on page one of Google, on autopilot across every platform, run my CRM in plain English, dispatch an AI executive team, host the whole thing on free infrastructure that fires while I sleep, and run every text, call, and voice agent across the entire business. The exact stack I used to go from invisible to ranked — zero coding experience and zero ad spend.

SYSTEM 1 · llms.txt
SYSTEM 2 · MARKDOWN MIRRORS
SYSTEM 3 · SITEMAPS + GSC
SYSTEM 4 · YOUTUBE AUTOPILOT
SYSTEM 5 · FULL WEBSITE BUILDER
SYSTEM 6 · CRM LEAD RECOVERY
SYSTEM 7 · CLAUDE CODE + GHL
SYSTEM 8 · AI EXECUTIVE TEAM
SYSTEM 9 · FREE INFRASTRUCTURE LAYER
SYSTEM 10 · MESSAGING LAYER

Built by @brycenwood.ai · May 2026

WHAT'S INSIDE · TAP TO JUMP

Nine Systems. One Playbook.

1

The AI Recommendation System (llms.txt)

Make ChatGPT, Claude, and Perplexity actively recommend your business when customers ask. The exact file and structure I used to get my shop quoted by name in AI answers. →

2

Markdown Mirrors

Give AI a clean, plain-text version of every page on your site so they can quote you faster, more accurately, and more often. The companion to llms.txt that almost nobody is doing yet. →

3

Sitemaps + Google Search Console

Tell Google exactly which pages matter, then see your actual ranking data — which keywords you rank for, what position you're in, and where the quick wins are. →

4

YouTube Autopilot

Auto-cross-post every Instagram Reel to YouTube Shorts with AI-optimized titles, descriptions, and tags built for YouTube's search algorithm. One prompt builds the entire system. →

5

Full Website Builder Prompt

The exact prompt I used to build a 40-page website in one afternoon. Swap in your business info, paste it into Claude Code, and get a complete site with SEO, schema, blog posts, and AI visibility built in. →

6

Automated CRM Lead Recovery →

Connect Claude Code to your CRM. It scans for missed leads every morning and drafts personalized follow-ups before you wake up. Set it on a cron job and never lose a lead again.

7

Claude Code + GHL →

Talk to your CRM in plain English. Pull stale leads, draft personalized follow-ups for all of them, audit broken automations, build workflows. The operating layer for everything else in this playbook.

8

AI Executive Team →

Build an AI C-suite for your business. 5 chief agents + specialists, each with a charter, dispatched in parallel from one prompt. The system that runs the other seven without you holding it all in your head.

9

The Free Infrastructure Layer (Cloudflare) →

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

10

The Messaging Layer (Twilio) →

Run every text, voice call, voicemail, and AI receptionist across your business through one plug-in. Pennies per message. Replaces ManyChat, Klaviyo SMS, RingCentral, and most of what you pay for inside GHL's SMS add-on. Includes A2P 10DLC registration, AI voice agents, and call recording + transcription.

HOW TO USE THIS PLAYBOOK

Each system has step-by-step prompts you paste directly into Claude Code. You can build all ten in a single afternoon. If you get stuck on any step or want the ready-made scripts, I share those inside the community at **brycenwood.com/community** — $97/mo for the first 200 founding members (price locked for life). The playbook alone is enough to ship everything.

SYSTEM 1

The AI Recommendation System

Make ChatGPT, Claude, and Perplexity recommend your business by name.

INTRO

What's up. I'm Brycen. I own a vehicle wrap company in Utah called Summit Wraps and Graphics. A couple months ago I typed "who does the best vehicle wraps in Utah" into ChatGPT and it recommended my shop. I didn't pay for it. I didn't run an ad. I just made my website readable to AI.

Here's the thing nobody is talking about yet: AI chat is replacing Google for a lot of searches. When someone asks ChatGPT for a plumber, an HVAC company, a contractor, a mechanic, that AI has to pull from somewhere. If your business shows up in those answers, you get the lead. If it doesn't, you don't exist.

This guide shows you exactly what I did, step by step, prompt by prompt. Copy it, swap in your business, drop it on your site. That's it.

WHAT YOU'LL BUILD

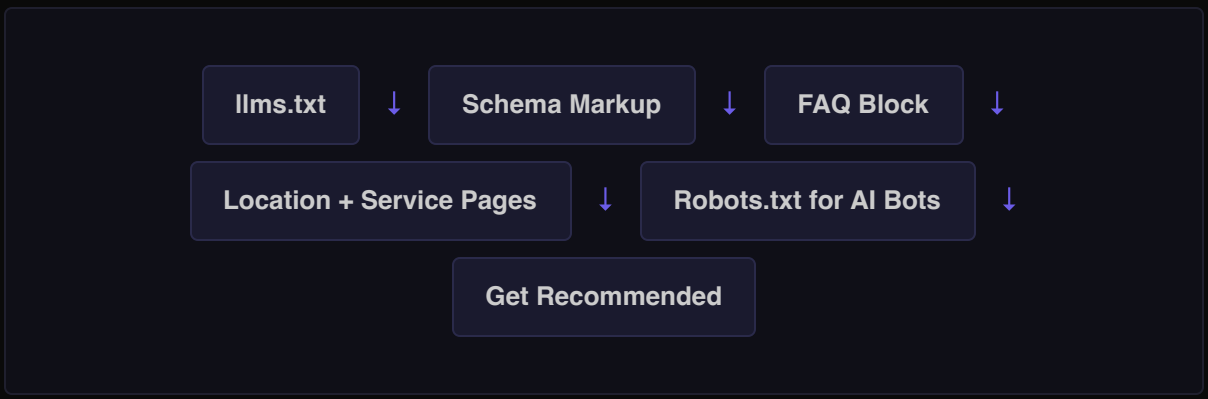
OVERVIEW

What You're Building

By the end of this guide, your website will have everything AI models need to recommend your business:

1. **An llms.txt file** — a plain text file that tells AI exactly what you do, where, and for how much
2. **Structured data (schema)** — machine-readable business info AI and Google both love

3. **A strong FAQ section** — direct Q&A pairs that AI quotes back to users
4. **Location + industry pages** — one page per city and per service so AI has something specific to cite
5. **AI crawler access** — robots.txt set up so GPTBot, ClaudeBot, and PerplexityBot can actually read your site



WHY THIS WORKS

AI models don't "search" the way Google does. They pull from pages that are clear, structured, and easy to summarize. Most business websites are bloated with marketing fluff and pop-ups. If you give AI a clean, direct source of truth, it will quote you over the noise.

PREREQUISITES

BEFORE YOU START

What You Need

- **Claude Code** installed on your computer (the CLI tool, not claude.ai)
- **Access to your website files** — whatever host you use (GitHub, Netlify, Squarespace, WordPress, Webflow, etc.). You need to be able to upload a new text file to the root of your site.
- **Basic info about your business** — services, pricing ranges, locations, hours, what makes you different. Have this in your head or in a notes doc before you start.
- **15–30 minutes** for the llms.txt + schema. A few hours if you want to do the full location/service pages.

HOW MUCH DOES THIS COST?

Zero extra dollars. The whole system is plain text files you drop on the site you already own. If you're on Claude Code's $20/month subscription you're already paying for it.

STEP 1

STEP 1

Create Your llms.txt File

This is the main file. It's a plain text document that sits at yourwebsite.com/llms.txt and tells AI exactly what your business is. Think of it like a one-page pitch written for a machine.

PASTE THIS INTO CLAUDE CODE

```
I need you to create an llms.txt file for my business. This is a plain text file that lives at the root of my website. It tells AI chat models (ChatGPT, Claude, Perplexity) exactly what my business does, so they can recommend me when someone asks about my industry.

Here are my business details:

Business Name: [YOUR BUSINESS NAME]
What I do: [ONE-SENTENCE DESCRIPTION]
Founded by / owners: [YOUR NAME]
Locations: [CITY, STATE — list all locations]
Phone: [YOUR PHONE]
Website: [YOUR URL]
Hours: [YOUR HOURS]
Service area: [LIST CITIES/COUNTIES YOU SERVE]

Services and pricing:
- [SERVICE 1]: [PRICE RANGE]
- [SERVICE 2]: [PRICE RANGE]
- [SERVICE 3]: [PRICE RANGE]
(add as many as you offer)

What makes me different: [2-3 SENTENCES ABOUT YOUR EDGE — materials, warranty, in-house vs. outsourced, experience, etc.]

Ideal customers / industries served: [WHO YOU WANT TO WORK WITH]

Key facts and credentials: [YEARS IN BUSINESS, # OF JOBS DONE, REVIEW
```

```
COUNT, CERTIFICATIONS, WARRANTY]

Now write the llms.txt file following this structure:
# [Business Name]

## About
(short paragraph)

## Services and Pricing
(bulleted list with real numbers — AI loves specific pricing)

## Locations
(bulleted list)

## Contact
(phone, website, social, hours)

## Service Area
(comma-separated list of cities)

## Key Facts
(bulleted list of credibility points)

## What Makes Us Different
(short paragraph)

## Frequently Asked Questions
(5-8 real Q&A pairs that a customer would actually ask — include
pricing, timing, warranty, common concerns)

Make it scannable. Use short sentences. Include specific numbers
wherever possible. No marketing fluff. Save it as llms.txt in my
website's root folder.
```

WHY PRICING MATTERS

AI models get asked "how much does X cost" constantly. If your llms.txt has real pricing ranges, you will get quoted. If it says "call for pricing" you will not. Give the AI something to say.

STEP 2

STEP 2

Upload It to Your Website

The file has to live at yourwebsite.com/llms.txt — not in a subfolder, not in a blog post, not behind a login. Root of the site. Test it in your browser after uploading by typing the URL directly.

PASTE THIS INTO CLAUDE CODE

```
I have a website hosted on [YOUR HOST —
GitHub/Netlify/Squarespace/WordPress/Webflow/etc.].

Walk me through exactly how to upload the llms.txt file we just
created to the root of my site. Give me step-by-step instructions for
my specific host. After I upload it, tell me how to confirm it's live
by visiting the URL directly.
```

COMMON GOTCHA

Some hosts (looking at you, Squarespace) make it hard to drop plain text files at the root. If yours is one of them, ask Claude Code for the workaround. There's always one — redirect, custom route, or switching to a different static host.

STEP 3

STEP 3

Add Schema Markup (Structured Data)

Schema is code AI and Google can read to understand exactly what your business is. It lives inside your website HTML, invisible to visitors, but very visible to machines.

PASTE THIS INTO CLAUDE CODE

```
I want to add JSON-LD schema markup to my homepage so AI and search
engines understand my business better.

Business details:
- Name: [YOUR BUSINESS]
- Type: [e.g., LocalBusiness, AutoRepair, Plumber, HVACBusiness,
Restaurant]
- Description: [ONE LINE]
```

```
- Address: [FULL STREET ADDRESS]
- Phone: [YOUR PHONE]
- Website: [YOUR URL]
- Hours: [YOUR HOURS]
- Price range: [$, $$, $$$]
- Services: [LIST]
- Review rating and count: [IF YOU HAVE GOOGLE REVIEWS]
- Lat/long: [IF YOU KNOW — otherwise skip]

Generate JSON-LD structured data using schema.org types:
1. LocalBusiness (or the most specific type for my industry)
2. FAQPage — with 5-8 common customer questions and answers
3. Service — one entry per main service I offer

Give me the code to paste inside the <head> section of my website.
Also explain what each block does so I can check it's accurate.
```

After you paste it in, test the schema at **validator.schema.org** and **search.google.com/test/rich-results**. Both free. They'll tell you if anything is broken.

STEP 4

STEP 4

Open the Door for AI Crawlers

AI bots like GPTBot (ChatGPT), ClaudeBot (Claude), and PerplexityBot have to be allowed to crawl your site. A lot of templates block them by default. You want them in.

PASTE THIS INTO CLAUDE CODE

```
Check the robots.txt file on my website at [YOUR URL]/robots.txt. I
want to make sure the following AI crawlers are explicitly ALLOWED
(not blocked):

- GPTBot (OpenAI / ChatGPT)
- ClaudeBot (Anthropic)
- PerplexityBot (Perplexity)
- Google-Extended (Google's AI training bot)
- CCBot (Common Crawl — a lot of models train on this)

If any are blocked, give me the updated robots.txt file I should
```

```
upload. Also add a line pointing to my sitemap.xml so everyone can
find my pages.
```

COUNTERINTUITIVE BUT TRUE

Every "SEO best practice" article from 2023 told you to block AI bots to "protect your content." Don't. You want AI to read your site. The whole point is to get quoted in AI answers. Let them in.

STEP 5

STEP 5

Add a Real FAQ Section to Your Homepage

AI pulls quotes directly from FAQ blocks. They're structured, they're direct, and they match how people ask questions. If your homepage has a good FAQ, AI will quote you word-for-word in its answers.

PASTE THIS INTO CLAUDE CODE

```
Write an FAQ section for my homepage. It should have 8–10 real
questions a potential customer would actually type into ChatGPT or
Google.

My business: [DESCRIBE IT]
My services: [LIST]
My pricing: [REAL RANGES]
My area: [CITY/STATE]

Rules:
– Every question should start with how, what, where, when, why, or do
– Every answer should be 2–4 sentences max, direct, include specific
numbers when possible
– Include at least one pricing question, one "how long does it take"
question, one "do you serve my area" question, and one "what makes you
different" question
– Write the answers so they can be quoted directly by AI — short,
factual, confident
– No marketing words: avoid "premier", "best-in-class", "industry-
leading", "cutting-edge"
```

```
Then wrap the whole thing in FAQPage schema markup so AI and Google
both index it properly.
```

STEP 6

STEP 6

Build Location + Service Pages (Optional but Powerful)

This is the step that took me from being mentioned to being the top recommendation. Create one page per city you serve and one page per core service. Each page is a clean, specific, AI-readable target.

PASTE THIS INTO CLAUDE CODE

```
I want to build location pages and service pages for my website that
rank in both Google and AI chat results.

My service area: [LIST OF CITIES YOU SERVE]
My core services: [LIST OF 3-6 MAIN SERVICES]

Generate two sets of pages:

LOCATION PAGES — one per city. URL format: /locations/[city-name]/
Each page should include:
- H1 with "[Service] in [City], [State]"
- Intro paragraph mentioning the city by name and my business by name
- 400-600 words covering: why local matters, services offered in that
city, specific neighborhoods or landmarks, pricing, contact info
- Embedded FAQ (3-5 questions) specific to that city
- Schema markup for LocalBusiness with the city's info
- Internal links back to the homepage and related service pages

SERVICE PAGES — one per core service. URL format: /services/[service-
name]/
Each page should include:
- H1 with the service name
- 600-800 words covering: what the service is, who it's for, what it
costs, how long it takes, what's included, the process step by step
- A "who this is for" section
- FAQ schema block
- Internal links back to the homepage and related services
```

```
Write real, useful content. Not keyword-stuffed garbage. Assume a real
customer will read these. Start with 3 location pages and 2 service
pages. We'll expand from there.
```

THE MULTIPLIER EFFECT

One llms.txt gets you in the door. Twenty location and service pages get you quoted. Every page is another chance for an AI to pull from your site instead of a competitor's. This is what took my traffic from ~7/day to ~150/day in two weeks.

STEP 7

STEP 7

Test It

Wait 3-7 days after everything is live (AI models need time to recrawl). Then test:

1. Open ChatGPT. Ask "who does the best [your service] in [your city]?"
2. Ask Perplexity the same question.
3. Ask Claude the same question.
4. Ask Google's AI Overview (appears at the top of normal Google searches) the same question.

If you show up in 1-2 of them early, you're on the path. If you show up in all 4, you've built a moat. Keep adding pages, keep updating your llms.txt as you add services, and check back every couple weeks.

PASTE THIS INTO CLAUDE CODE

```
Help me audit how well my AI SEO is working. Check [YOUR URL] and give
me a report on:

1. Is my llms.txt live and does it have everything important?
2. Is my schema markup valid?
3. Is robots.txt allowing AI bots?
4. Does my homepage have a real FAQ section with schema?
5. How many location pages do I have? Are they thin or real content?
6. What's missing that top AI-recommended sites in my industry have?
```

```
Give me a prioritized list of fixes, most impactful first.
```

RESULTS

MY NUMBERS

What Happened for Summit Wraps

- **Old site:** ~7 real visitors per day
- **New site (two weeks after launch):** ~100-150 real visitors per day
- **Traffic increase:** ~1,400%
- **ChatGPT result:** actively recommends Summit Wraps when asked about vehicle wraps in Utah
- **Google Search Console:** ranking page 1 for "car wraps near me" within weeks
- **Cost:** $0 in ads, $0 to an agency, ~1 week of building with Claude Code
- **Coding experience used:** zero

> REAL TALK
>
> This isn't a hack and it's not going to stay this easy. Right now, almost nobody is doing this. That's the opportunity. In 12 months every decent website will have an llms.txt and schema and location pages. Do it now, while the door is still wide open.

TIPS

TIPS

Things I Learned Building This

1. **Specificity beats polish.** Real prices, real cities, real numbers. AI will quote specifics. It will ignore vague marketing copy.
2. **Update llms.txt every time your business changes.** New service? New price? New location? Update the file. It's literally a text doc.
3. **Don't block AI bots, even "to save on bandwidth."** You want the exposure more than you want the server cycles.

4. **One clean page beats five bloated ones.** A 500-word location page with real info outperforms a 3,000-word SEO monster stuffed with keywords.
5. **AI cares about FAQs more than paragraphs.** Questions and answers get quoted. Paragraphs get ignored. Add FAQs everywhere.
6. **Check your Google Search Console.** You'll start seeing impressions for queries you never targeted. That's AI-adjacent traffic showing up.
7. **Ask AI about yourself regularly.** "Best [service] in [city]" — do this weekly. It's free feedback on what's working.

SYSTEM 2

Markdown Mirrors

Give AI a clean version of every page on your site to read.

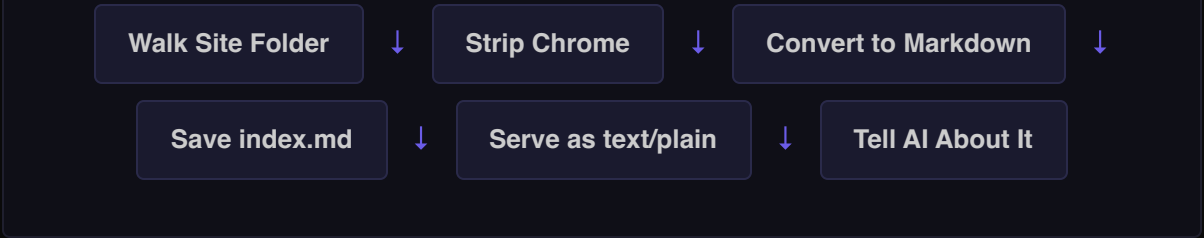
OK so this is the second piece of the AI visibility puzzle. The llms.txt file you built in System 2 tells AI models *what* your business is. But when AI tries to read individual pages on your site to quote you, it has to wrestle through navigation, popups, scripts, cookie banners, and all the chrome a normal website has for humans. Markdown mirrors solve that.

A markdown mirror is a clean, plain-text version of every page on your website that lives at a predictable URL — just add `/index.md` to any page. Anthropic's docs site does this. Stripe's docs site does this. Almost no small businesses do this yet. That's the opportunity.

OVERVIEW

What You're Building

One Python script that walks your website folder and, for every HTML page, generates a sibling `index.md` file containing only the actual content. Plus a small Netlify config tweak so those `.md` files render as plain text in browsers (instead of forcing a download), and an update to your `llms.txt` so AI knows the mirrors exist.



WHY THIS STACKS WITH LLMS.TXT

llms.txt is the pitch — the elevator overview for your whole business. Markdown mirrors are the per-page detail. When AI cites you, it can quote from the right page directly. You want both. The two together turn your site into a fully AI-readable surface.

BEFORE YOU START

What You Need

- **Claude Code** installed (the CLI tool, not claude.ai)
- **A static website** hosted on Netlify, Vercel, GitHub Pages, Cloudflare Pages, or any host that lets you upload a config file. If you're on Squarespace or Wix, you'll need to ask Claude Code for the workaround for your specific host.
- **Your website code in a folder** you can hand to Claude Code. If your site is in a GitHub repo, that's perfect.
- **Python 3 installed** with the `beautifulsoup4` and `markdownify` packages (Claude Code will install these for you if you ask).

TIME + COST

End to end this took me about an hour, including testing. Cost: $0 ongoing. The script is one Python file you run whenever your site updates. You can wire it into a pre-commit hook later if you want it fully automated.

STEP 1

Build the Generator Script

Open Claude Code in your website project folder. Paste this prompt. It will write a Python script that walks every `index.html` file in your site, strips the chrome, and writes a clean markdown version next to it.

PASTE THIS INTO CLAUDE CODE

```
I want to build markdown mirrors for every page on my website. The
goal is to give AI tools (ChatGPT, Claude, Perplexity) a clean version
of each page they can read without wrestling with HTML, scripts, or
chrome.
```

```
Write me a Python script that:

1. Walks my website folder and finds every index.html file (skip 404s
and any /thanks/ pages that are noindex)
2. For each page, parses the HTML with BeautifulSoup
3. Strips out: nav, footer, scripts, styles, noscripts, chat widgets,
GHL/HubSpot widgets, iframes, and any element with a class matching
nav, footer, cta-split, or starting with ghl
4. Drops empty div/span wrappers that have no text content
5. Converts the remaining body HTML to clean markdown using the
markdownify package
6. Cleans the markdown output: collapses 3+ blank lines, strips
standalone "01" "02" step numbers, removes bullet separator
characters, removes empty image alt tags
7. Writes the result to {page_dir}/index.md with frontmatter at the
top:

---
title: [page title from <title>]
description: [meta description]
url: [canonical URL]
last_updated: [today's date]
---

8. Prints a summary at the end showing how many pages were generated

Save it as generate_markdown_mirrors.py in my scripts folder. Make it
re-runnable so I can rerun it whenever my site updates. After you
write it, run it once and show me the output.
```

WHY FRONTMATTER MATTERS

The frontmatter at the top of each markdown file gives AI extra context — title, description, the canonical URL. When AI quotes the page, it knows where the content came from. Don't skip it.

SHORTCUT

If you'd rather skip building the script yourself, the working Python generator is available inside the community at **brycenwood.com/community**. First 200 founding members lock in $97/mo for life. Paste it in, run it, done.

STEP 2

Configure Your Host to Serve .md as Plain Text

By default, most web hosts treat `.md` files as a download — when you visit one in a browser, it downloads instead of displaying. That's bad for two reasons: humans can't preview them, and some AI fetchers stumble on the content type. You want them to render inline as plain text.

PASTE THIS INTO CLAUDE CODE

```
I'm hosted on [YOUR HOST — Netlify / Vercel / Cloudflare Pages /
GitHub Pages / Squarespace / etc].

I just generated markdown mirror files at {page}/index.md across my
site. I need every .md file to be served with Content-Type: text/plain
so they render in the browser instead of triggering a download.

Show me exactly which config file I need to edit on my host and the
exact lines to add. Then explain how to test it after deploy.

For Netlify, the answer is the _headers file. For Vercel, it's
vercel.json. For Cloudflare Pages, it's _headers also. Walk me through
whichever one applies to me.
```

FOR NETLIFY USERS SPECIFICALLY

Add this to your `_headers` file:

```
/*.md
  Content-Type: text/plain; charset=utf-8
  Cache-Control: public, max-age=3600
  X-Robots-Tag: index, follow
```

STEP 3

Tell AI the Mirrors Exist

You already have an llms.txt file from System 2. Add a section to it that lists every markdown mirror URL so AI fetchers know where to find them.

PASTE THIS INTO CLAUDE CODE

```
Open my llms.txt file. Add a new section called "Markdown Mirrors"
right above the "What Makes Us Different" section. Inside that new
```

```
section, list every index.md URL on my site. The format should be:

## Markdown Mirrors (Clean AI-Readable Versions)
Every page on this site has a plain markdown mirror. Add /index.md to
any URL to get the clean content without navigation, scripts, or
layout chrome.

- https://yourdomain.com/index.md
- https://yourdomain.com/about/index.md
- https://yourdomain.com/services/index.md
(etc, one per page)

Generate the full list automatically by reading my website folder.
Skip any noindex pages (thanks pages, 404s).
```

STEP 4

Test It

After you push the changes and your site redeploys, run through this test:

1. **Open one of your mirror URLs in a browser** — for example `yourdomain.com/business-wraps/index.md`. It should render as clean plain text, not download.
2. **Curl the URL from terminal** — `curl -I https://yourdomain.com/business-wraps/index.md`. The Content-Type header should say `text/plain`.
3. **Paste a mirror URL into ChatGPT** — ask "what does this business do?" ChatGPT will fetch the URL and return a clean summary. If it nails it, your mirrors work.
4. **Paste the same URL into Claude or Perplexity** — same test. Verify all three platforms can read it cleanly.

> IF THE TEST FAILS
>
> If the .md still downloads instead of displaying, the Content-Type header didn't take. Check that `_headers` is in your site's root and the format is exact (the indentation matters on Netlify). Redeploy and test again.

STEP 5

Keep It Updated

Whenever you update a page on your site, you'll want to regenerate that page's markdown mirror so AI is reading the latest version. Two ways to handle this:

Option A — Manual rerun

Just run `python3 generate_markdown_mirrors.py` whenever you update content. Takes about 2 seconds. Commit + push. Done.

Option B — Automated via pre-commit hook

PASTE THIS INTO CLAUDE CODE

```
I want my markdown mirrors to regenerate automatically whenever I
commit changes to my website. Set up a pre-commit hook that:

1. Runs generate_markdown_mirrors.py every time I git commit
2. Adds any newly generated/updated .md files to the commit
automatically
3. Doesn't fail the commit if the script has any non-fatal warnings

Walk me through the .git/hooks/pre-commit setup for my repo. Make it
so I never have to think about it again.
```

MY RESULTS

What Happened After I Shipped This

- **27 markdown mirrors** generated across my site in about 2 seconds
- **Total time to build + ship:** ~1 hour from "what's a markdown mirror" to live in production
- **Cost:** $0 (just the Claude Code subscription I already had)
- **ChatGPT, Claude, and Perplexity** can now read each page in one clean fetch with no HTML noise
- **The video about it** hit my biggest reach yet within hours of posting

REAL TALK

This is one of those "almost nobody is doing this yet" plays. In 12 months it'll be standard. Right now you can ship it in an hour and be early. That's the whole game with

AI visibility — be early, be specific, give the machines something clean to quote.

TIPS

Things I Learned Building This

1. **Keep the markdown clean.** If your generator leaves visual chrome like "01" "02" or bullet separator characters in the output, AI will get tripped up. Strip aggressively.
2. **Frontmatter helps AI.** The title/description/URL block at the top of each .md file gives the AI context. Don't skip it.
3. **Don't change the URL pattern later.** Once you decide on `/page/index.md` or `/page.md`, stick with it. Changing breaks any AI fetcher that already learned where to look.
4. **Update llms.txt every time you add a new page.** Otherwise AI won't know the new mirror exists. Or wire that into your generator script.
5. **Test with all three: ChatGPT, Claude, Perplexity.** They have different fetchers and sometimes one will work while another won't. You want all three reading you.
6. **Don't gate the .md files with login or paywall.** The whole point is for AI to read them freely. If you need gating, gate the HTML version, not the markdown.

SYSTEM 3

Sitemaps + Google Search Console

Tell Google which pages matter. Then see your actual ranking data.

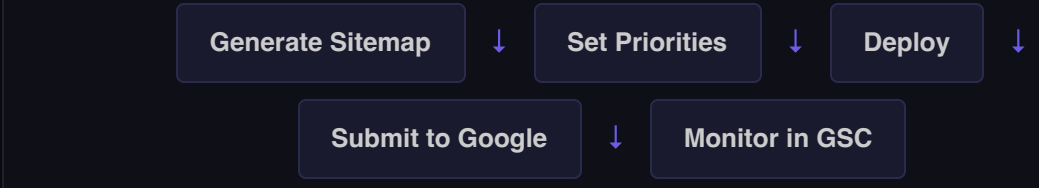
This is the third piece. llms.txt tells AI what your business is. Markdown mirrors let AI read your pages clean. A sitemap tells *Google* which pages exist, which ones matter most, and when they were last updated. Without it, Google is guessing. With it, you're handing Google a map of your entire site with priorities marked.

Then Google Search Console lets you see exactly what's happening — which keywords you're ranking for, what position you're in, and where to focus next. Most small businesses never set this up. That's the gap.

OVERVIEW

What You're Building

An XML sitemap that lists every important page on your site with priority scores and update frequencies. Plus a Google Search Console setup so you can see your actual ranking data — which keywords are bringing people to your site, what position you're in, and how your click-through rate compares.



WHY THIS STACKS WITH LLMS.TXT + MIRRORS

llms.txt is for AI chatbots. Markdown mirrors are for AI page readers. The sitemap is for Google's crawler. Together, these three files cover every way a machine discovers and reads your business online. You're not choosing between traditional SEO and AI visibility — you're doing both with three text files.

BEFORE YOU START

What You Need

- **Claude Code** installed (the CLI tool, not claude.ai)
- **Your website code in a folder** that Claude Code can access
- **A Google account** for Search Console (free)
- **Access to your domain's DNS** (Cloudflare, GoDaddy, Namecheap, etc.) — needed for one verification step

TIME + COST

Sitemap generation: 5 minutes. Google Search Console setup: 10 minutes. Total: 15 minutes. Cost: $0. You'll start seeing ranking data within 2-3 days.

STEP 1

Generate Your Sitemap

Open Claude Code in your website project folder. Paste this prompt. It will create an XML sitemap that lists every page on your site with the right priorities.

PASTE THIS INTO CLAUDE CODE

```
I need an XML sitemap for my website. Generate a sitemap.xml file
that:

1. Lists every important page on my site (skip 404 pages, thank-you
pages, and any noindex pages)
2. Sets the homepage priority to 1.0
3. Sets main service pages to 0.9
4. Sets other pages (about, contact, gallery) to 0.8
5. Sets blog posts or secondary pages to 0.7
6. Uses today's date for lastmod on all pages
```

```
7. Sets changefreq to "weekly" for the homepage and "monthly" for
everything else
8. Follows the standard XML sitemap protocol at sitemaps.org

Save it as sitemap.xml in the root of my website folder. After you
create it, show me the full file so I can review it before deploying.
```

WHY PRIORITIES MATTER

The priority score (0.0 to 1.0) tells Google which pages you consider most important. Your homepage and main service pages should be highest. This doesn't guarantee Google will rank them higher, but it tells the crawler where to spend its time. Most auto-generated sitemaps set everything to 0.5 — that's like telling Google "nothing matters more than anything else."

STEP 2

Deploy and Verify

Push your sitemap.xml to your live site. Then verify it's accessible.

PASTE THIS INTO CLAUDE CODE

```
My sitemap.xml is deployed. Help me verify it's working:

1. Fetch https://[MY-DOMAIN]/sitemap.xml and confirm it returns valid
XML
2. Check that every URL in the sitemap actually resolves (no 404s)
3. Count the total pages listed
4. Show me any issues

Also check if I have a robots.txt file. If I do, make sure it includes
a Sitemap: line pointing to my sitemap. If I don't have a robots.txt,
create one that:
- Allows all crawlers
- Explicitly allows AI crawlers (GPTBot, ClaudeBot, PerplexityBot,
Google-Extended)
- Points to my sitemap URL
```

STEP 3

Set Up Google Search Console

This is how you see your actual Google ranking data. Most business owners have never looked at this — they just guess whether SEO is working. GSC shows you exactly.

1. **Go to** `search.google.com/search-console`
2. **Click "Add property"** and choose "Domain" (not "URL prefix")
3. **Enter your domain** (e.g. yourbusiness.com — no https, no www)
4. **Google will give you a TXT record** to add to your DNS. Copy it.
5. **Go to your DNS provider** (Cloudflare, GoDaddy, etc.) and add the TXT record
6. **Go back to GSC** and click "Verify." It may take a few minutes for DNS to propagate.
7. **Once verified, go to Sitemaps** in the left sidebar and submit your sitemap URL

> IF YOU GET STUCK ON DNS VERIFICATION
>
> This is the one step that trips people up. If you're not sure where your DNS is hosted, ask Claude Code: "Where is my domain's DNS managed?" and give it your domain name. It can usually figure it out. Or just search "[your registrar] add TXT record" — every provider has a guide.

STEP 4

Read Your Data

After 2-3 days, GSC will start showing data. Here's what to look for:

1. **Go to Performance** in the left sidebar
2. **Check all four boxes** at the top: Total clicks, Total impressions, Average CTR, Average position
3. **Look at the Queries table** — these are the keywords people are searching when they find you
4. **Sort by Impressions** — high impressions + low clicks = your biggest opportunity (people see you but don't click)
5. **Sort by Position** — anything between position 5-15 is a "quick win" because you're close to page one

> THE QUICK WIN FORMULA

Find keywords where you rank position 5-15 with high impressions. These are keywords where Google already thinks you're relevant — you just need a small push to get onto page one. That push is usually: better meta title, better meta description, and a clean page structure. Ask Claude Code to optimize those specific pages for those specific keywords.

IF YOU WANT HELP READING YOUR GSC DATA

Post a screenshot of your Performance tab in the community (**brycenwood.com/community**) and I'll tell you exactly which keywords to target first. First 200 founding members lock in $97/mo for life. Seeing the data is easy — knowing what to do with it is where people get stuck.

MY RESULTS

What Happened After I Shipped This

- **Position 6 for "car wraps near me"** — the most valuable keyword in my industry, on page one of Google
- **Position 1 for "car wrap places near me"** and **"car wrap services"**
- **131 clicks from Google** in the first month of tracking
- **334 total keywords** my site now ranks for (most I didn't even know about)
- **Total time to set up:** 15 minutes for the sitemap + GSC, zero dollars

REAL TALK

The sitemap alone didn't do this. The llms.txt + markdown mirrors + sitemap together is what moved the needle. Google is starting to reward sites that are clean, structured, and AI-readable. If you've already built Systems 1 and 2 from this playbook, the sitemap is the last piece that ties it all together for Google's crawler. Three files. That's the whole system.

TIPS

Things I Learned

1. **Update your sitemap when you add pages.** If you add a new service page or blog post, regenerate the sitemap. Otherwise Google won't know about it for weeks.
2. **Don't obsess over position daily.** Rankings fluctuate. Check weekly, not hourly. The trend line matters more than any single day.
3. **GSC data is delayed 2-3 days.** What you see today is from 2-3 days ago. Don't panic if numbers look weird — wait for the data to catch up.
4. **Your meta title and description matter more than you think.** GSC will show you CTR (click-through rate). If you have high impressions but low CTR, your title and description aren't compelling enough. Ask Claude Code to rewrite them.
5. **The robots.txt + sitemap combo is standard practice.** But adding llms.txt and markdown mirrors on top of it is what puts you ahead. Most businesses stop at the sitemap. You're not stopping there.

SYSTEM 4

YouTube Autopilot

Auto-cross-post every video to YouTube with AI-optimized metadata.
One prompt builds it.

This is the system that changed how I think about content. You're already filming videos. You're already posting on Instagram and TikTok. But if you're not on YouTube, you're leaving the biggest search engine in the world on the table.

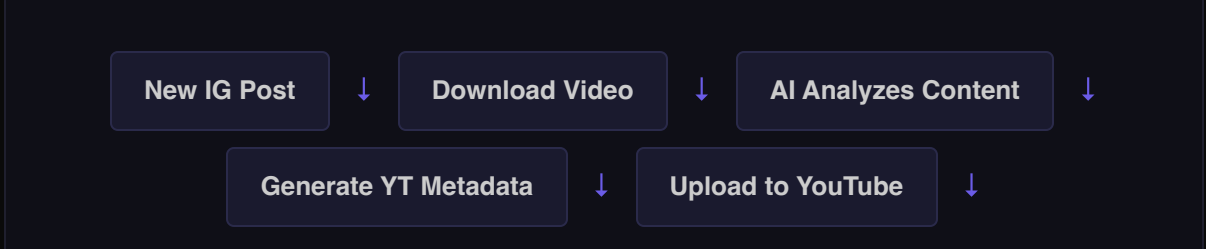
The problem is nobody wants to manually upload to another platform every day. And if you just paste your Instagram caption onto YouTube, it dies. YouTube's algorithm is search-based. Instagram's is engagement-based. Same caption doesn't work on both.

This system solves both problems. It watches your Instagram, grabs every new video, uses AI to generate a completely new title, description, and tags optimized for YouTube search, and uploads it as a Short. Automatically. You set it up once and never think about YouTube again.

OVERVIEW

What You're Building

Two Python scripts that work together. The first watches your Instagram via the official API, downloads new videos, and generates YouTube-optimized metadata using Claude. The second uploads them to your YouTube channel with the optimized titles, descriptions, and tags, and drops a CTA comment on every video.



Post CTA Comment

WHY THIS IS DIFFERENT FROM REPURPOSE.IO OR MANUAL CROSS-POSTING

Apps like Repurpose.io just mirror your video with the same caption. That's not cross-posting, that's copy-pasting. YouTube's algorithm needs searchable titles, keyword-rich descriptions, and relevant tags to surface your content. This system generates all of that automatically using AI that actually analyzes your video content and researches what's ranking on YouTube for that topic.

BEFORE YOU START

What You Need

- **Claude Code** installed (the CLI tool)
- **An Instagram business or creator account** with access to the Instagram Graph API (Meta developer account, free)
- **A YouTube channel** (free, takes 2 minutes to create if you don't have one)
- **A Google Cloud project** with the YouTube Data API enabled (free tier covers everything)
- **An Anthropic API key** for Claude (usage-based, costs less than $0.01 per video)

TIME + COST

Setup: about 1-2 hours including API credentials. Ongoing cost: under $1/month for the AI metadata generation. The YouTube API and Instagram API are both free. Once it's running, it takes zero time because it's fully automated.

STEP 1

Set Up YouTube API Access

You need a Google Cloud project with the YouTube Data API enabled. This gives your script permission to upload videos and post comments on your channel.

PASTE THIS INTO CLAUDE CODE

```
I need to set up YouTube API access so I can upload videos
programmatically. Walk me through:

1. Creating a Google Cloud project (or using an existing one)
2. Enabling the YouTube Data API v3
3. Creating OAuth 2.0 credentials (Desktop app type)
4. Downloading the credentials.json file
5. Where to save it in my project

I want to upload YouTube Shorts and post comments on my own channel. I
don't need to access anyone else's channel.
```

THE OAUTH DANCE

The first time you run the upload script, it will open a browser window asking you to log into Google and authorize the app. Pick the correct YouTube channel on the consent screen. After that, it saves a token and never asks again (it auto-refreshes). If you have multiple YouTube channels, make sure you select the right one.

STEP 2

Set Up Instagram Graph API Access

Your script needs to read your Instagram posts to know when you've posted new videos. This uses Meta's official Graph API.

PASTE THIS INTO CLAUDE CODE

```
I need to set up Instagram Graph API access to read my posts
programmatically. Walk me through:

1. Creating a Meta developer account and app (if I don't have one)
2. Adding Instagram Graph API to the app
3. Generating a long-lived access token for my Instagram account
4. Testing the token to make sure it can read my media

My Instagram handle is [YOUR HANDLE]. I only need to read my own
posts, not manage other accounts.
```

```
Save the config to a JSON file with the access token and my Instagram
user ID.
```

STEP 3

Build the Cross-Post System

This is the main prompt. It builds the entire system: the Instagram watcher, the AI metadata generator, and the YouTube uploader. One prompt, two scripts, fully automated.

PASTE THIS INTO CLAUDE CODE

```
I want to build a system that automatically cross-posts my Instagram
Reels to YouTube Shorts with AI-optimized metadata. Build me two
Python scripts:

SCRIPT 1: Instagram to Queue
- Check my Instagram account for new video posts using the Graph API
- Download any new videos it finds
- For each video, read my Instagram caption and use the Anthropic API
(Claude) to generate:
  - A YouTube-optimized title (searchable, keyword-rich, under 80
characters, front-load the main keyword)
  - A YouTube description rewritten from scratch for search (NOT a
copy of the IG caption). Include relevant keywords, a hook, key
takeaways, and a CTA section at the bottom with links to my playbook
and social profiles
  - 15-20 YouTube tags mixing exact-match search phrases, related
topics, and broad category tags
- Save everything to a JSON queue file
- Track which posts have already been processed so it skips duplicates
on re-run

SCRIPT 2: Queue to YouTube
- Read the JSON queue
- Upload each video as a YouTube Short (public, category: Education)
- Post a pinned comment with a CTA on each upload
- Handle YouTube quota limits gracefully (pause if exceeded)
- Save the YouTube video ID back to the queue so we know what's
uploaded

Both scripts should be re-runnable and safe to call from a cron job.
Add a --dry-run flag to both. The Instagram script should have a --
backfill flag to reprocess everything.
```

```
My Anthropic API key is in my .env file as ANTHROPIC_API_KEY.
My Instagram config is at _config/instagram.json.
My YouTube credentials are at credentials.json with token at
_config/youtube_token.json.
```

WANT THE WORKING SCRIPTS?

The exact Python scripts I use are available inside the community at **brycenwood.com/community**. First 200 founding members lock in $97/mo for life. Paste the prompt above to build your own version, or grab the battle-tested ones from the community.

STEP 4

Automate It

The scripts work manually, but the whole point is to never think about YouTube again. Set up a cron job (Linux) or launchd agent (Mac) to run them on a schedule.

PASTE THIS INTO CLAUDE CODE

```
I want to automate my YouTube cross-posting system so it runs every 6
hours without me doing anything. Set up:

1. A scheduled task that runs the Instagram-to-queue script every 6
hours
2. Immediately after, runs the YouTube uploader script
3. Logs output to a file so I can check what happened
4. Doesn't run if the previous run is still going

I'm on [Mac/Linux]. Use launchd for Mac or cron for Linux. Make sure
it runs even if I'm not logged in.
```

MY RESULTS

What Happened After I Shipped This

- **5,000+ views on YouTube** across 12 videos without ever opening YouTube to upload a single one

- **Every 6 hours** the system checks for new Instagram posts and uploads them automatically
- **Each video gets a unique YouTube-optimized title** instead of a pasted Instagram caption
- **Total time spent on YouTube after setup:** zero
- **Setup time:** about 2 hours including API credentials
- **Ongoing cost:** under $1/month for the AI metadata generation

> REAL TALK
>
> The biggest unlock here isn't the automation. It's that YouTube is a search engine, not a feed. Your Instagram videos have a 24-48 hour shelf life. YouTube Shorts can get discovered months later through search. Every video you post is a long-term asset on YouTube and a short-term play on Instagram. This system makes sure you get both without any extra work.

TIPS

Things I Learned

1. **YouTube titles and Instagram captions are different languages.** Instagram captions are conversational ("Comment MIRROR and I'll DM you"). YouTube titles are searchable ("How to Make Your Website Readable by AI"). The AI rewrite handles this but understand why it matters.
2. **YouTube tags have a 500-character total limit.** Don't generate 30 long-tail tags. 15-20 concise tags is the sweet spot.
3. **The YouTube Data API has a daily quota.** You get 10,000 units/day for free. Each upload costs ~1,600 units. That's about 6 uploads per day, which is more than enough for cross-posting.
4. **Post a CTA comment on every video.** YouTube comments with links actually work because YouTube doesn't suppress them like Instagram does. Use this for your playbook, community, or website link.
5. **Don't delete and re-upload.** If you want to update metadata on an existing video, use the API to update in place. This preserves your view count, comments, and watch history.
6. **YouTube Shorts get discovered through search for months.** Unlike Instagram where a post dies in 48 hours, YouTube Shorts surface in search results indefinitely.

Every video is a long-term asset.

SYSTEM 5

Full Website Builder Prompt

Build a complete 40-page website in one afternoon. One prompt. Zero code.

THE PROBLEM

Most AI-built websites are invisible

Anyone can ask AI to build a website. You'll get a homepage and a contact form and it'll sit on the internet and do nothing. No SEO. No schema. No sitemap. No AI visibility. Google doesn't know it exists. ChatGPT can't read it. It's a digital business card that nobody sees.

I spent months figuring out what actually makes an AI-built website show up in Google, get recommended by ChatGPT, and bring in leads on its own. Then I put all of it into one prompt.

THE PROMPT

Paste this into Claude Code

This prompt is the bones. It tells Claude Code exactly what to build and how to structure it so your site actually gets found. But the more real information you feed it about your business, the better the results. Don't just fill in the brackets and hit enter -- give it everything. Your services, your story, your customers, what makes you different, your pricing, your location, examples of your work. The more context Claude Code has, the more specific and useful every page will be.

Replace the bracketed fields below to get started. Then keep the conversation going -- tell it more about your business and watch the site get better with every detail you add.

PASTE THIS INTO CLAUDE CODE

```
I need you to build me a complete website for my business.

Business name: [YOUR BUSINESS NAME]
What we do: [ONE SENTENCE - what you sell or offer]
Who we serve: [YOUR IDEAL CUSTOMER]
Location: [CITY, STATE or "online"]
What makes us different: [1-2 SENTENCES]

Build me:
- Homepage with hero section, services overview, testimonials section, and contact form
- Services page with individual service detail pages
- About page with our story
- Contact page with a working form
- Blog with 5 SEO-optimized articles targeting what our customers actually search for
- Case study page with 1-2 examples of our work
- Pricing page (if applicable)
- FAQ page with schema markup
- Privacy policy and terms of service
- llms.txt so AI tools like ChatGPT, Claude, and Perplexity can read and recommend our site
- Markdown mirrors (index.md) for every page so AI reads clean content
- XML sitemap
- Schema markup (JSON-LD) on every page - LocalBusiness, Service, FAQPage, Article, BreadcrumbList
- Meta titles and descriptions optimized for clicks on every page
- Mobile responsive, fast loading, modern design
- robots.txt that allows all AI crawlers

Deploy-ready. No frameworks. Pure HTML and CSS. Every page should be its own folder with an index.html inside it.
```

WHAT THIS BUILDS

Here's what you get from one prompt

This isn't just a website. It's a website that's built to be found. And the more you tell Claude Code about your business after the initial build, the deeper it goes -- rewriting blog posts with your actual expertise, building FAQ pages from real customer questions, creating case studies from your real projects.

- **30-40+ pages** depending on your services and blog topics
- **SEO on every page** - meta titles, descriptions, heading structure, internal linking
- **Schema markup** - Google Rich Results for your business, services, FAQ, and articles

- **AI visibility** - llms.txt + markdown mirrors so ChatGPT and Perplexity can read and recommend you
- **XML sitemap** - tells Google exactly which pages to index
- **Blog posts** - targeting real keywords your customers search for
- **Mobile responsive** - looks good on every device
- **No monthly fees** - deploy free on Cloudflare Pages, Netlify, or GitHub Pages

DEPLOY IT

Get it live in 5 minutes

Once Claude Code builds your site, you need to put it on the internet. All of these are free:

Option A: Cloudflare Pages (recommended)

> PASTE THIS INTO CLAUDE CODE
>
> ```
> Help me deploy this website to Cloudflare Pages. Walk me through:
> 1. Creating a Cloudflare account (if I don't have one)
> 2. Connecting my GitHub repo
> 3. Setting up the build
> 4. Connecting my custom domain
> ```

Option B: Netlify

> PASTE THIS INTO CLAUDE CODE
>
> ```
> Help me deploy this website to Netlify. Walk me through connecting my
> GitHub repo and setting up my custom domain.
> ```

Both options are free, auto-deploy when you push changes, and include SSL (the lock icon in the browser).

MAKE IT YOURS

Customize after the first build

The prompt gives you the structure, but the magic happens when you keep feeding Claude Code more about your business. The first build gets you 90% of the way there. Then you refine by telling it what to change, what to add, what to go deeper on:

PASTE THIS INTO CLAUDE CODE

```
I want to customize my website. Here's what I want to change:
- [DESCRIBE WHAT YOU WANT DIFFERENT - colors, layout, new pages,
different copy, etc.]

Keep everything else the same. Don't break the SEO, schema, or AI
visibility files.
```

That last line is important. It tells Claude Code to preserve all the invisible infrastructure (schema, sitemap, llms.txt, markdown mirrors) while changing the visual design. Most people rebuild from scratch when they want changes. You just describe what you want different.

THE SECRET TO A GREAT AI-BUILT WEBSITE

The prompt is the skeleton. Your business knowledge is the muscle. The more you tell Claude Code -- your actual customer stories, your real process, the questions people always ask you, why you started the business -- the more it sounds like YOU wrote every page, not an AI. Don't hold back. Dump everything you know into the conversation.

REAL RESULT

I used this exact approach to build brycenwood.com - 40+ pages, deployed in one afternoon, ranking on Google within days. Blog posts, case studies, industry pages, guides, pricing, contact form, full SEO, AI visibility. Zero code written. Zero dollars spent on a developer.

SYSTEM 6

Automated CRM Lead Recovery

Never miss a lead again. AI scans your CRM every morning, finds who slipped through, and writes personalized follow-ups before you wake up.

THE PROBLEM

Leads come in. Nobody follows up.

Every business loses money the same way: a lead fills out your form at 11 PM, your team is asleep, and by morning it's buried under 30 other things. Studies show that responding within 5 minutes makes you **21x more likely** to close the deal. Most businesses respond in 24-48 hours. Some never respond at all.

This system connects Claude Code directly to your CRM. Every morning, it scans for new leads that came in overnight, checks if anyone followed up, and drafts personalized emails based on each lead's industry. You wake up, review the drafts, and hit send. Zero leads slip through.

STEP 1

Connect Claude Code to your CRM

This works with GoHighLevel (GHL), but the same concept applies to any CRM with an API (HubSpot, Salesforce, etc). You need your API key and your location/account ID.

PASTE THIS INTO CLAUDE CODE

```
I want to connect you to my GoHighLevel CRM so you can read my
contacts and pipeline.

Here's my setup:
- API key: [YOUR GHL API KEY - find this in Settings > Business
Profile > API Key]
```

```
- Location ID: [YOUR LOCATION ID - find this in Settings > Business
Profile]

Create a config file at _config/ghl.json that stores these
credentials.
Then test the connection by pulling my 5 most recent contacts and
showing me their names and when they were added.
```

WHERE TO FIND YOUR GHL API KEY

In GHL, go to Settings > Business Profile > scroll to "API Key." Copy the full key (starts with "pit-"). Your Location ID is on the same page. If you're using HubSpot or another CRM, ask Claude Code to help you find your API credentials.

STEP 2

Build the lead scanner

This script checks your CRM for any new contacts that came in since the last scan and checks whether anyone on your team responded. It flags the ones that fell through the cracks.

PASTE THIS INTO CLAUDE CODE

```
Build me a Python script called lead_scanner.py that:

1. Connects to my GHL CRM using the config in _config/ghl.json
2. Pulls all contacts added in the last 24 hours
3. For each contact, checks if there are any outbound messages (email,
SMS, or call)
4. Outputs a table showing:
   - Contact name
   - Email and phone
   - When they came in
   - Source (form, Google, Instagram, etc)
   - Whether we responded (yes/no)
5. Highlights any contact with NO outbound response

Save the results to _data/leads/daily_scan.json so we have a history.
```

STEP 3

Add personalized email drafting

This is where it gets powerful. For every lead that didn't get a response, Claude Code writes a personalized follow-up based on their industry, what they submitted, and your services.

PASTE THIS INTO CLAUDE CODE

```
Now add a drafting step to lead_scanner.py. For every contact that has
NO outbound response:

1. Look at their company name, email domain, and any form data they
submitted
2. Determine their industry (roofing, HVAC, landscaping, restaurant,
etc)
3. Draft a short, personalized follow-up email that:
   - References their industry specifically
   - Mentions a pain point relevant to their business
   - Keeps it under 100 words (they read on their phones)
   - Sounds human, not templated
   - Ends with a clear next step (reply, call, or book a time)
4. Save drafts to _data/leads/email_drafts.json
5. Show me all the drafts for review before anything gets sent

IMPORTANT: Never send anything automatically. Draft only. I review and
approve every email before it goes out.
```

STEP 4

Set it up to run every morning automatically

Now you automate it with a cron job. This runs the scanner every morning at 7 AM so the drafts are waiting for you when you wake up.

PASTE THIS INTO CLAUDE CODE

```
Set up a cron job (or macOS LaunchAgent) that runs lead_scanner.py
every morning at 7:00 AM.

Requirements:
- Runs the full scan + draft pipeline
- Logs output to _data/logs/lead_scanner.log
- If there are new un-responded leads, save the drafts
- If there are zero new leads, log "No new leads overnight" and exit
quietly
- Send me a summary notification (optional: via email, Slack, or just
log it)
```

```
Show me how to verify it's running and how to check the logs.
```

MAC VS LINUX VS WINDOWS

On Mac, Claude Code will create a LaunchAgent (runs even when Terminal is closed). On Linux, it uses crontab. On Windows, it uses Task Scheduler. Just tell Claude Code what OS you're on and it'll set up the right one.

STEP 5

Review and send

Every morning, you do three things:

1. **Check the drafts** - open _data/leads/email_drafts.json or ask Claude Code to show you
2. **Edit anything that needs tweaking** - change a line, add a detail, adjust the tone
3. **Send the approved ones** - either through your CRM or ask Claude Code to send via your email

PASTE THIS INTO CLAUDE CODE

```
Show me this morning's lead scan results and email drafts. For each draft, show me the lead's info and the proposed email side by side so I can review them quickly.
```

Total time: 5 minutes. Every lead gets a personalized response within hours of submitting. That alone puts you ahead of 90% of your competitors.

REAL RESULT

I run this exact system for my vehicle wrap business. It scans GHL every morning, finds overnight leads, and drafts industry-specific emails. Before this, leads would sit for 2-3 days before anyone noticed. Now every lead gets a personalized response before 8 AM. The system paid for itself the first week.

SYSTEM 7

Claude Code + GHL

Talk to your CRM in plain English. The operating layer for the leads everything else generates.

System 6 gives you a morning lead-recovery cron. System 7 gives you the full bridge. Once Claude Code is connected to GHL at the API level, you can ask it to do anything you would normally have to click through menus for — not just morning lead drafts. Pull lists, audit broken workflows, build new ones, draft 50 personalized follow-ups in one prompt, build dashboards on demand.

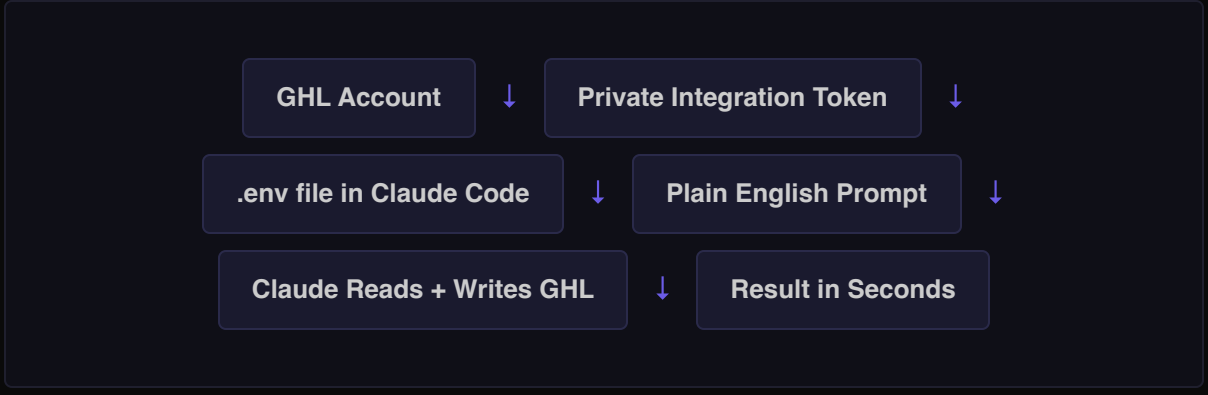
Here is the thing about GHL. Most local service businesses pay for it and use about 10% of what it does. The UI is brutal. Building a workflow takes 30 clicks. Pulling a list of leads who haven't been followed up with takes you through three different screens. Most owners never crack the magic.

This system is the operating layer for the other six in this playbook. Your llms.txt brings AI traffic. Your sitemap brings Google traffic. Your website turns visits into leads. GHL captures them. System 6 catches the overnight stragglers. And System 7 is how you actually run the whole thing without a team.

OVERVIEW

What You're Building

A direct connection between Claude Code (the AI you talk to like a coworker) and GHL (the CRM running your business). Once it is set up, you can ask Claude Code to read from your GHL, write to your GHL, draft messages, find broken automations, build dashboards, walk you through workflow setup — anything you would normally have to click through the UI for.



> **HOW THIS STACKS WITH SYSTEM 6**
>
> System 6 uses a small config file (`_config/ghl.json`) to drive a single morning cron. System 7 sets up a full `.env` + helper file pattern that any prompt or script in your project can reuse. Build System 6 first if you only want overnight lead recovery. Build System 7 when you want Claude Code to operate every part of GHL in plain English.

BEFORE YOU START

What You Need

- **Claude Code** installed (the CLI tool, not claude.ai)
- **A GHL sub-account**. Any tier works. You do not need agency-level access.
- **5 minutes** to grab a Private Integration Token from GHL settings
- **Python 3** on your machine. Claude Code installs any packages it needs.

> **TIME + COST**
>
> End to end this took me about 10 minutes the first time. Cost: $0. Once it is set up, every prompt you run after that is just talking to Claude Code like any other conversation.

STEP 1

Generate a Private Integration Token in GHL

GHL needs a way to know that Claude Code is allowed to talk to it. That is what a Private Integration Token (PIT) does. Think of it as a key that only your account holds.

1. Log into your GHL sub-account

2. Click your profile (bottom-left) then **Settings**
3. Scroll the left sidebar and click **Private Integrations**
4. Click **Create New Integration**
5. Name it: `Claude Code`
6. Select scopes: check every contact, opportunity, workflow, conversation, and custom field scope. You can always tighten later.
7. Click **Create**
8. Copy the token that appears. You will only see it once.

> **TREAT THIS TOKEN LIKE YOUR PASSWORD**
>
> Do not share it. Do not commit it to GitHub. We will store it in a .env file (and Claude Code will set up your project to ignore that file automatically).

If GHL has moved a menu since this was written, paste this into Claude Code and it will walk you through the current path:

> **PASTE THIS INTO CLAUDE CODE**
>
> ```
> Walk me through getting a Private Integration Token from my
> GoHighLevel sub-account, with current 2026 UI screenshots if needed. I
> need a token with full access to contacts, opportunities, workflows,
> conversations, and custom fields. Tell me exactly which menus to
> click. If you are unsure of the current path, tell me what to search
> for inside GHL settings.
> ```

STEP 2

Connect Claude Code to Your GHL

Now we tell Claude Code where to find your GHL. Open Claude Code in any project folder, or make a new one called `ghl-ops` just for this. Paste the prompt below. Claude Code will set up your environment, build a small Python helper, and run a test to confirm the connection works.

> **PASTE THIS INTO CLAUDE CODE**
>
> ```
> Set up a connection between this project and my GoHighLevel sub-
> account.
> ```

```
1. Create a .env file with these two variables:
   GHL_API_KEY=[I will paste my Private Integration Token here]
   GHL_LOCATION_ID=[my sub-account location ID]

2. Add .env to .gitignore so the token never gets pushed to GitHub.

3. Build a small Python helper file called ghl_client.py with:
   - A function that loads the .env values
   - A function that lists contacts (with filters for tag, last
activity, opportunity status, and date range)
   - A function that reads a contact's full conversation history
   - A function that sends a contact an SMS (but only when I
explicitly ask)
   - A function that adds or removes a tag on a contact
   - A function that lists all workflows and pipelines in the sub-
account

4. Use the GHL API v2 with the Authorization: Bearer header and
Version: 2021-07-28 header.

5. After you build it, run a test that pulls 5 contacts from my GHL
and prints their names. Confirm the connection works. If it fails,
debug and fix it before stopping.
```

WHERE TO FIND YOUR LOCATION ID

In GHL: profile menu, then Settings, then Business Profile. The Location ID is in the URL of that page (the long string after `/location/`). Or just ask Claude Code: "Help me find my GHL Location ID."

GHL API GOTCHA TO KNOW ABOUT

The contacts endpoint sortBy parameter has to be snake_case (`date_added`) not camelCase (`dateAdded`). It is not in the GHL changelog. If your test fails with a 422, that is probably why. Tell Claude Code and it will fix it.

STEP 3

Pull Your Stale Leads in One Sentence

This is the moment most service biz owners realize what is actually possible. With the connection from Step 2 working, paste this prompt:

PASTE THIS INTO CLAUDE CODE

```
Pull every contact in my GHL that:
- Was created in the last 90 days
- Has no opportunity attached, OR has an opportunity stuck in the first stage
- Has not been messaged by me or a teammate in the last 30 days

Sort by created date (oldest first). Output as a clean table with: name, phone, original first message, days since last activity, and any tags on the contact.

Do not message anyone yet. Just show me the list.
```

WHAT JUST HAPPENED

This list normally takes 20+ clicks across three GHL screens to assemble — and most owners never figure out how to filter for it. Now it takes one sentence. Welcome to the new layer.

STEP 4

Draft a Personalized Follow-Up for Every Stale Lead

System 6's morning cron drafts emails for overnight leads. Here is how you do the same thing on demand for any list of contacts — pulled from any filter, with SMS or email, all in one prompt.

PASTE THIS INTO CLAUDE CODE

```
For each contact in the stale leads list from Step 3, draft a follow-up SMS that:
- Sounds casual and like a real person, not a sales bot
- References what they originally asked about (use their first message in GHL conversations)
- Does not pressure them
- Ends with a soft question to restart the conversation
- Is under 160 characters

Show me the drafts in a table next to each contact. I will review before anything sends. After I approve, send the approved batch with a 30-second delay between each so it does not look automated.
```

ALWAYS REVIEW BEFORE SENDING

Do not auto-send AI-drafted messages without a human eyeball, especially the first few times. The 30 seconds you spend reviewing is worth it. After a few rounds, you will trust the patterns and you can let Claude Code send the approved batch on its own.

STEP 5

Audit Your GHL Top to Bottom

This is the prompt I run on every account I touch. It is how I found 5 critical bugs in my own sales pipeline that had been silently dropping leads for weeks.

PASTE THIS INTO CLAUDE CODE

```
Audit my GHL sub-account end to end. Look for:

1. Workflows that are "Published" but have not fired in 30+ days
(probably broken or no triggers matching)
2. Contacts created in the last 90 days with no opportunity attached
and no follow-up activity
3. Opportunities sitting in any stage longer than 21 days with no
notes or activity
4. Custom fields that exist but have no data populated for more than
50% of contacts (probably orphaned)
5. Pipelines with stages that have no contacts, ever
6. Tags applied to fewer than 3 contacts (probably one-off mistakes)
7. Contacts with no first name or no phone (data hygiene issues)

Output a prioritized list of issues, sorted by likely impact (lost
revenue first). Do not fix anything yet. Just show me what is broken.
```

REAL RESULT

I ran this exact audit on my Summit Wraps GHL on May 4, 2026. It found 5 critical bugs in our sales pipeline that had been silently dropping leads for weeks. Two were dead automations (workflows that were "Published" but pointing to deleted custom fields). Three were stalled opportunities no one had touched. Total recovery: 47 leads moved back into active follow-up. None of it would have been caught by clicking around in the UI.

STEP 6

Build a Speed-to-Lead Workflow With Claude Code as Your Guide

Speed to lead — texting a new lead within 30 seconds of them filling out a form — is the single highest-impact automation a service business can run. Most never set it up because the GHL workflow builder is exhausting. Have Claude Code walk you through the exact clicks.

PASTE THIS INTO CLAUDE CODE

```
I want to build a speed-to-lead workflow inside the GHL workflow
builder UI. Walk me through every click step by step.

Workflow:
- Trigger: form submission on any inbound web form
- Action 1: send the contact an SMS within 30 seconds: "Hey [first
name], got your message. What's the best number to reach you at?"
- Action 2: wait 30 minutes
- Action 3: if they have not replied, send a second SMS: "Just making
sure this came through. When's a good time to call?"
- Action 4: add the tag "speed-to-lead" so I can track it

Tell me exactly which menu to click first, what to name each step,
what fields to fill, and which dropdowns to choose. Account for any
GHL UI quirks you know about. After I confirm it is built, ask me to
test it by submitting my own form and tell me what to look for.
```

WHY THIS IS THE HIGHEST ROI STEP IN THIS WHOLE PLAYBOOK

Studies show that texting a lead within 5 minutes vs 30 minutes of inquiry increases the chance of converting them by up to 21x. Most service businesses miss this because the manual setup is hard. With Claude Code guiding you, you set it up once and never think about it again.

STARTER PROMPT LIBRARY

Three More Prompts to Run Today

Once the connection works, you will start asking Claude Code to do things you never realized were possible. Here are three more starter prompts to get the muscle memory

going.

PASTE THIS INTO CLAUDE CODE

```
Find every customer who paid me money in the last 12 months but has not booked again. Draft each one a personalized re-engagement text that references the service they paid for last time. Send 5 a day so it does not look spammy.
```

PASTE THIS INTO CLAUDE CODE

```
Read every conversation in my GHL from the last 14 days. Tell me which inbound customer messages I never responded to. Sort by deal size if I have it on the contact, otherwise by most recent first.
```

PASTE THIS INTO CLAUDE CODE

```
Build me a daily revenue dashboard from my GHL. Pull total opportunities by stage, total opportunities won this week vs last week, average days in pipeline, and top 5 lead sources by conversion rate. Email it to me every morning at 7am.
```

WHERE THIS GOES NEXT

This is the bridge. Every system in this playbook gets more powerful the moment it is connected to GHL through Claude Code. Your llms.txt knows which services convert. Your YouTube cross-poster tags new viewers as leads. Your website's contact form fires a workflow before the page even reloads. The next chapter is System 8 — once you have one Claude Code talking to your CRM, you can have a whole team of them, each with a defined role, all working in parallel.

SYSTEM 8

The AI Executive Team

Build an AI C-suite for your business. 5 chief agents + specialists, each with a charter, dispatched in parallel from one prompt. The system that runs the other seven without you holding it all in your head.

THE PROBLEM

One AI is good. A team of them is unreasonable.

By System 7, you have Claude Code talking to your CRM. You can pull stale leads in one sentence, audit your whole pipeline, build workflows. It is wild — until you try to do all of it on the same day. Then you remember why you wanted help in the first place. There are too many parts of a business for one prompt to hold at once.

Real companies solve this with an executive team. A CMO who only thinks about marketing. A CFO who only thinks about money. A COO who keeps operations running. Each one has a defined scope. Each one knows when to escalate. None of them have to hold the whole business in their head — they just hold their slice and hand off the rest.

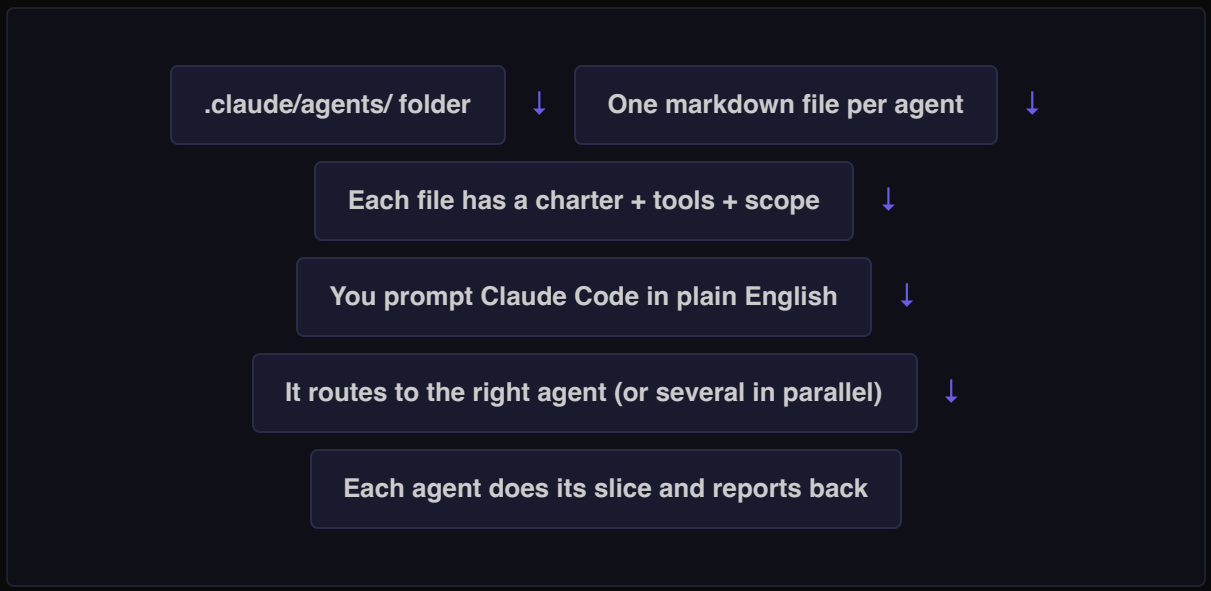
System 8 builds that team for you, in Claude Code. Five chief agents, each with their own charter. Specialists under them for the work the chiefs don't do directly. You dispatch one of them with a sentence and they go to work. You dispatch all of them and they go to work in parallel. While you sleep. While you film. While you go to the gym.

OVERVIEW

What You're Building

An `.claude/agents/` folder inside your project. Each agent is a single markdown file with a name, a description of when to use it, the tools it can access, and a system prompt

that defines its role. Claude Code reads the folder automatically and routes your prompts to the right agent — or spawns several of them at once.



HOW THIS STACKS WITH THE REST OF THE PLAYBOOK

Systems 1-5 build your visibility. Systems 6 + 7 connect Claude Code to your CRM so it can act. System 8 turns that one Claude Code into an organized team. Your CRO agent watches the leads from System 6. Your CMO agent owns the SEO from Systems 1-3. Your COO agent makes sure the website (System 5) and YouTube (System 4) keep running. They each report to your Chief of Staff agent — and your Chief of Staff reports to you.

BEFORE YOU START

What You Need

- **Claude Code** installed (the CLI tool, not claude.ai)
- **An existing project folder** where you already use Claude Code (the same folder where you built System 6 or 7 works)
- **10-20 minutes** to build your first chief agent and dispatch it
- That is it. No new accounts, no new APIs, no new costs. Subagents are built into Claude Code.

TIME + COST

End to end: about 30 minutes the first time to set up your first 2 chief agents. After that, every new agent takes 5-10 minutes. Cost: roughly $1/month extra in Anthropic API

usage from running the team — most of your cost is still you talking to Claude Code directly.

STEP 1

Stand up the agents folder

Open Claude Code in your project. The first move is creating the folder Claude Code looks at to find your team.

PASTE THIS INTO CLAUDE CODE

```
I want to build an AI executive team inside this project using Claude Code subagents.

Create a .claude/agents/ folder in this project. Then create a README inside that folder explaining:
- That each .md file in this folder is one agent
- The required frontmatter for an agent file (name, description, tools)
- The pattern for writing the system prompt as the body of the file
- A note that subagents run in their own context window, so they don't pollute the main conversation

Show me the README when done.
```

WHY A FOLDER OF MARKDOWN FILES

This is the actual format Claude Code uses to register subagents. Each .md file becomes a callable agent. You can edit them in any text editor, version them in git, and copy them between projects. No platform lock-in.

STEP 2

Build your first chief agent — Chief of Staff

The Chief of Staff is the agent you start with because everything else routes through it. Its job is to hold your strategic priorities across every part of your business and tell you (or

other agents) what to work on right now. It is the closest thing to having a real human chief of staff who actually keeps the map in their head for you.

PASTE THIS INTO CLAUDE CODE

```
Create my first subagent inside .claude/agents/.

File: .claude/agents/chief-of-staff.md

Required frontmatter:
- name: chief-of-staff
- description: Strategic router and weekly review owner. Holds
priorities across all parts of my business. Use for "what should I
work on right now" decisions, weekly reviews, and cross-priority
calls.
- tools: Read, Edit, Write, Bash, Glob, Grep, WebFetch

System prompt body should establish:
1. The agent's role (chief of staff for my business)
2. The scope of what it owns (strategic priorities, weekly reviews,
cross-priority decisions)
3. The scope of what it does NOT own (it does not run marketing
campaigns, it does not touch the CRM directly, it does not write code
— it routes those to specialist agents)
4. The format it should respond in (always lead with the recommended
next action, then the reasoning, never bury the answer)
5. When it should escalate back to me (anything involving money over
$X, anything legal, anything personal)

Also: the system prompt should make this agent talk to me like a homie
who has my best interest in mind, not like corporate consultant.
Direct, casual, no fluff.

Build the file. Then read it back to me so I can confirm.
```

THE CHARTER IS THE MAGIC

An agent's charter is the system prompt. It defines what the agent owns, what it doesn't, how it should respond, and when to escalate. If you spend an extra 5 minutes making the charter sharp, the agent gets dramatically better. If the charter is vague, the agent will be vague back. Treat it like a job description for a real hire.

STEP 3

Add your other four chiefs

Now repeat the pattern for your other four executive agents. CMO, CRO, COO, CFO. Each one gets their own .md file in `.claude/agents/` with a frontmatter block and a charter.

PASTE THIS INTO CLAUDE CODE

```
Create the other four chief agents in .claude/agents/. Same format as
chief-of-staff.md. For each one, write a sharp charter that defines
scope, format, and escalation rules.

1. .claude/agents/chief-marketing-officer.md
   - Owns: brand strategy, content calendar, ad performance, audience
growth, partnerships
   - Does NOT own: closing deals, billing, operations
   - Tools: Read, Edit, Write, Bash, Glob, Grep, WebFetch, WebSearch

2. .claude/agents/chief-revenue-officer.md
   - Owns: pipeline integrity, conversion rate, sales cycle time,
win/loss analysis, hot inbound
   - Does NOT own: product delivery, brand decisions, capital
allocation
   - Tools: Read, Edit, Write, Bash, Glob, Grep, WebFetch

3. .claude/agents/chief-operating-officer.md
   - Owns: calendar integrity, vendor uptime, SOPs, dashboard health,
integration QA, HR ops
   - Does NOT own: strategic priorities, marketing, finance
   - Tools: Read, Edit, Write, Bash, Glob, Grep, WebFetch

4. .claude/agents/chief-financial-officer.md
   - Owns: P&L per business, AR, capital allocation, pricing strategy,
owner pay policy, Profit First allocation
   - Does NOT own: marketing spend decisions (CMO partners on those),
operational vendors (COO owns)
   - Tools: Read, Edit, Write, Bash, Glob, Grep, WebFetch

Build all four. Then list every agent now in .claude/agents/ so I can
confirm the team is staffed.
```

YOU CAN CHANGE THE NAMES

If you are not running multiple businesses, you do not need a CRO and a CMO and a COO. You might just need a "marketing agent" and a "sales agent" and a "ops agent."

The names are yours. The pattern is the same. Define the scope sharply, define what it does not do, give it the tools it needs.

STEP 4

Add the specialists

Chiefs do strategic work. Specialists do executional work. Below your CRO, you might have a "closer agent" that drafts hot inbound replies. Below your CMO, a "content agent" that runs your post pipeline. Below your CFO, a "bookkeeper agent" that reconciles your accounting daily.

You do not need 13 specialists tomorrow. You need 1 specialist for whichever lane is most painful today.

PASTE THIS INTO CLAUDE CODE

```
I want to add a specialist agent under one of my chiefs. The most
painful unowned work in my business right now is [DESCRIBE THE WORK —
e.g. "drafting hot inbound replies", "reconciling QBO transactions
daily", "auditing my SEO rankings weekly"].

Pick a sensible name for this specialist. Create
.claude/agents/[specialist-name].md with:
- name + description (when to use)
- tools (only what it actually needs — read-only for audit agents,
write access for content/messaging agents)
- A sharp charter that explains:
  - The exact scope of what this agent owns
  - Which chief it reports to
  - The format of its outputs (always show me the draft before
sending, never auto-send messages, etc.)
  - When to escalate to its chief vs back to me directly

After you build it, dispatch this new specialist with a small test
task to confirm it works.
```

READ-ONLY BY DEFAULT

When you build a new specialist, give it the smallest tool set it needs. An audit agent only needs Read + Glob + Grep. A reporting agent only needs Read + Bash. Only give Write

or send-message tools to agents whose entire job is to produce drafts you approve. This is the same principle as least-privilege access for human employees.

STEP 5

Dispatch one agent

Now you actually use the team. The simplest dispatch is asking Claude Code to run a single agent on a single task.

PASTE THIS INTO CLAUDE CODE

```
Spawn the chief-of-staff agent. Give it this brief:

GOAL: Tell me what I should focus on this week.

CONTEXT: Read my latest dossier in _data/reports/daily/, look at my
open tasks in _data/personal/tasks.json, and check git log for what
shipped in the last 7 days.

CONSTRAINTS: Maximum 3 priorities. Each one needs a "why now" reason.
Skip vanity work.

DEFINITION OF DONE: A 3-bullet list with rationale, sent to me in this
conversation.

Run it. Show me the report when it comes back.
```

THE DISPATCH CONTRACT

GOAL / CONTEXT / CONSTRAINTS / DEFINITION OF DONE is the format that makes agent dispatches reliable. Every time you brief an agent, give it those four things. No agent does its best work from a one-line prompt. Treat them like sharp humans who need a written brief.

STEP 6

Spawn the team in parallel

This is the moment the system gets unreasonable. You ask Claude Code one question. It dispatches multiple agents at the same time. They each do their slice in their own context window. They report back. You read the synthesis.

PASTE THIS INTO CLAUDE CODE

```
I want a snapshot of where my business stands right now. Spawn my CMO, CRO, COO, and CFO in parallel. Each one writes a 5-bullet brief from their lane.

CMO brief: state of brand + content + audience growth this week. Top 1 thing to push on.
CRO brief: state of the pipeline. Hot leads. Stalled deals. Top 1 thing to close.
COO brief: state of operations. What is slipping. Top 1 thing to firm up.
CFO brief: state of money. Top 1 thing to chase or pay.

Run them in parallel, not in sequence. After all four return, synthesize the four briefs into a single "what matters most this week" recommendation. Show me each brief AND the synthesis.
```

WHY PARALLEL MATTERS

If you ran these one at a time, the conversation would burn through tokens carrying every previous brief into the next agent's context. By spawning them in parallel, each agent gets its own clean context, runs its work, and returns a tight summary. You only see the summaries — not the work that produced them. That is how the system stays cheap and fast even with a real team.

STEP 7

Add scope rules + safety

The single biggest risk with an agent team is an agent doing something you did not ask for — sending a message you would not have sent, deleting a file you needed, calling an API you did not authorize. The fix is writing scope rules into every charter and adding hard safety gates for high-blast-radius actions.

PASTE THIS INTO CLAUDE CODE

```
Audit every agent in .claude/agents/. For each one, do two things:

1. Read the system prompt and check that it explicitly states:
   - The exact scope of what the agent OWNS (positive scope)
   - The exact scope of what the agent DOES NOT OWN (negative scope)
   - When the agent must escalate back to me (e.g., any send-to-customer action, any deletion, any spend over $X)

2. Read the tools list in the frontmatter and check that it follows the least-privilege principle. Audit agents should be Read-only. Drafting agents should NEVER auto-send. Only agents whose entire job is to send a message I approved should have send-message tools.

For any agent missing scope or with overly-broad tools, propose a fix. Do not modify anything yet. Just show me the audit and the proposed fixes. I will approve before any changes.
```

DEFAULT TO DRAFT-ONLY

Make every agent that sends messages, edits production data, or spends money draft-only by default. The agent prepares the action, hands it to you for review, and you click send. After 30-60 days of trust building, you can let specific agents auto-execute on specific narrow tasks. Earn the auto-pilot, do not assume it.

STARTER PROMPT LIBRARY

Three more dispatches to run today

Once your team is built, you will start asking it to do things you did not realize were possible. Three to start with.

PASTE THIS INTO CLAUDE CODE

```
Spawn my CRO. Pull every lead in my GHL that has gone cold (no inbound or outbound activity in 14+ days). For each one, write a one-sentence reason it probably went cold and a one-sentence reactivation hook tailored to that lead's original inquiry. Show me the table. I will pick which to send.
```

PASTE THIS INTO CLAUDE CODE

```
Spawn my CMO. Audit my last 30 days of content. Group posts by hook
style and topic. Tell me the 1 hook style that is over-performing, the
1 topic I have been avoiding, and the 3 specific posts I should make
in the next 7 days based on that data.
```

PASTE THIS INTO CLAUDE CODE

```
Spawn my CFO. Read every QBO transaction from the last 30 days.
Categorize anything uncategorized. Flag any vendor charge over $100
that wasn't in last month's pattern. Show me the YTD P&L per business
with what changed vs last month. Do not modify QBO. Read-only audit.
```

REAL RESULT

I run a 5-chief + 13-specialist team across my 3 businesses (vehicle wraps, meal prep, personal brand). Total cost: about $1/month extra in API usage. They do morning audits, draft hot inbound replies, reconcile my accounting, audit my SEO, and surface what I am avoiding. The biggest unlock is not any single agent — it is that I no longer hold all 3 businesses in my head every day. The Chief of Staff agent does. I just ask "what should I focus on right now" and it tells me. Built every charter from a couch with zero coding experience.

SYSTEM 9

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.

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.

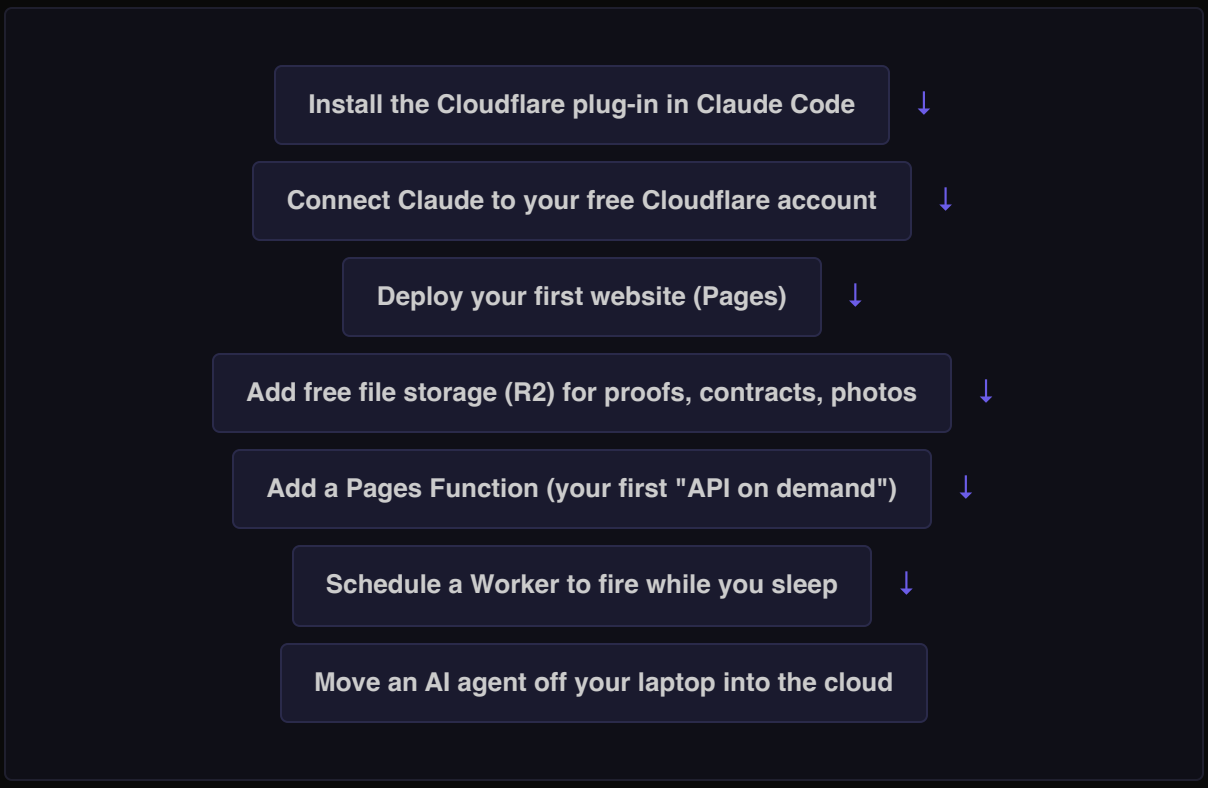
System 9 is that infrastructure. Cloudflare. The same network that runs Shopify checkouts, Disney+, and most of the modern internet. They give it away at small scale because most owners never even discover it exists. Inside Claude Code, there is a plug-in that connects directly to it. Once it is installed,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

OVERVIEW

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.

BEFORE YOU START

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.

STEP 5

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.

STEP 6

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.

STEP 7

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.

STARTER PROMPT LIBRARY

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.

SYSTEM 10

The Messaging Layer

Run every text, voice call, voicemail, and AI receptionist across your business through one plug-in. Pennies per message. Replaces ManyChat, Klaviyo SMS, RingCentral, and most of what you pay for inside your CRM's SMS add-on.

THE PROBLEM

Your business is duct-taped together with messaging tools.

Walk through any service business and the comms stack looks the same. ManyChat for Instagram DMs ($15-79/mo). Klaviyo SMS for marketing texts ($30-100/mo). RingCentral or OpenPhone for the business line ($20-30/mo). Your CRM's SMS add-on for customer notifications ($30-50/mo). A separate WhatsApp tool because your CRM doesn't do it well ($25/mo). Maybe a CallRail-style number for tracking ($45/mo). When you want a customer to be reachable through every channel, you cobble it together with Zapier and pray nothing breaks.

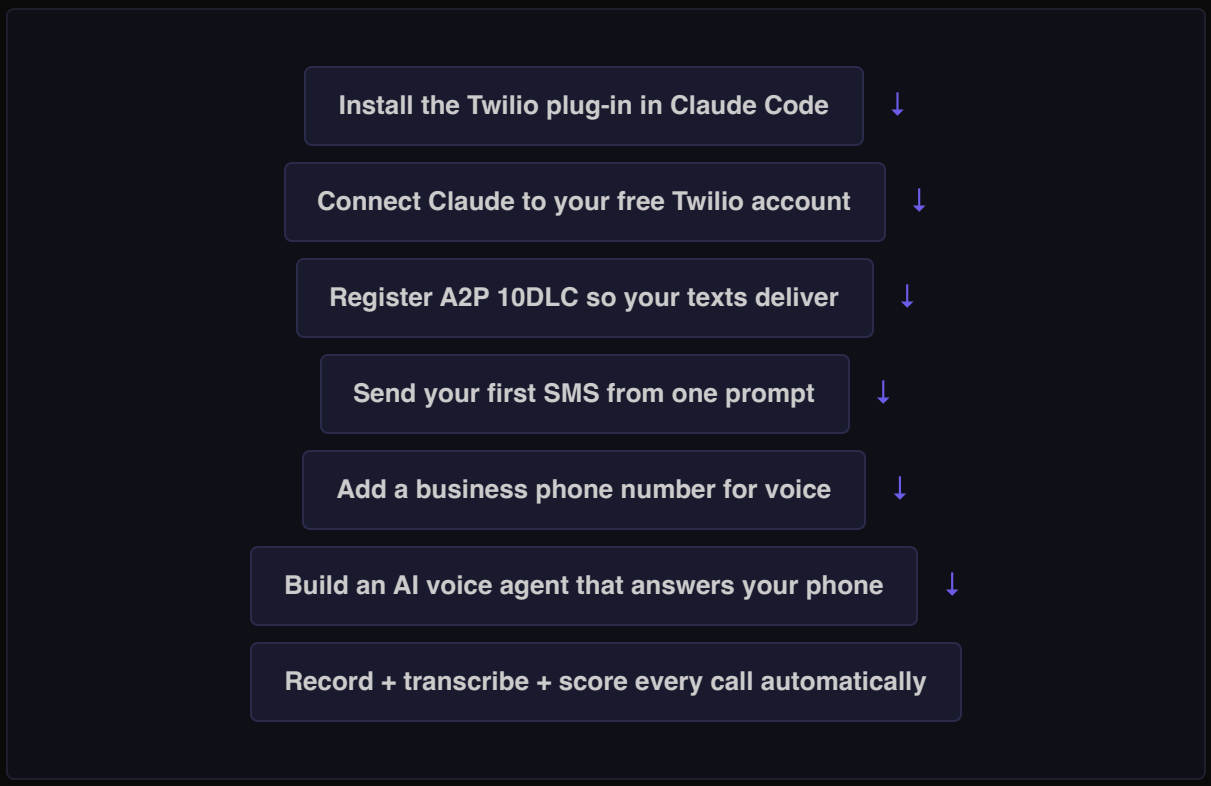
That is $150-330 a month in messaging tools. All of them — every single one — are wrappers around the same underlying platform. Twilio. The actual layer that carries the SMS, the voice call, the WhatsApp message, the RCS card, the verification code. The wrappers pay Twilio a fraction of a cent per message and charge you a flat $30-100 a month. You are paying the markup on the wrapper for features the underlying platform already gives you for free.

System 10 is the underlying platform. Twilio. Inside Claude Code there is a plug-in that connects directly to it. Once it is installed, every text, every call, every voicemail, every AI receptionist, every appointment reminder across your entire business runs through one layer. Pennies per message. No monthly subscription. No more "which tool do I check for that message?"

OVERVIEW

What You're Building

A single messaging layer that handles every outbound + inbound text, call, and voice interaction across your business. Customer order confirmations (Gainz Train uses this). Reactivation campaigns to dormant leads (Summit Wraps uses this). AI receptionist that answers the phone when you cannot (the next build in my own stack). Appointment reminders. WhatsApp business chat. Branded RCS messages. Phone-number fraud checks before you send to a new lead. A2P 10DLC compliance registration so your texts actually deliver. All of it from one plug-in, in plain English, inside Claude Code.



HOW THIS STACKS WITH THE REST OF THE PLAYBOOK

System 7 connects Claude Code to your CRM. System 8 builds your AI executive team. System 9 puts the whole thing on free infrastructure that runs 24/7. System 10 is the layer that lets all of those agents and workflows actually *reach* a customer. Your CRO agent (System 8) drafts a reactivation message — System 10 sends it. Your scheduled Worker (System 9) finds a missed lead at 5am — System 10 texts them back. Your AI receptionist runs on Cloudflare (System 9) but talks to customers through Twilio (System 10). Build System 10 after System 9 — it is the voice and hands of every other system.

BEFORE YOU START

What You Need

- **Claude Code** installed (the CLI tool, not claude.ai)
- **A free Twilio account.** Sign up at `twilio.com`. You get a small trial balance to play with. Add a credit card after you have tested — pay-as-you-go pricing, no monthly minimum.
- **A US-based business with a real EIN or sole-prop registration.** A2P 10DLC registration (Step 3) requires this. If you are not in the US, the registration step is different but the rest of the playbook is identical.
- **Your business contact info ready** — legal name, address, EIN or DUNS, point-of-contact email, sample messages you plan to send. A2P registration asks for all of this up front.
- **The Twilio plug-in** enabled in Claude Code (we install this in Step 1).
- **System 9 already built** (optional but recommended). The cleanest way to run scheduled SMS campaigns is from a Cloudflare scheduled Worker that talks to Twilio. If you skipped System 9, the prompts below still work — they just run on your laptop instead of in the cloud.

> TIME + COST
>
> End to end: about 60 minutes to install the plug-in, connect Twilio, send your first SMS, and add a phone number for voice. A2P 10DLC registration takes 5-10 minutes to submit but the actual carrier approval takes 1-3 weeks (you can keep building while you wait). After it is wired, every new campaign or workflow is 5-15 minutes. Cost: free to install, plus actual usage. SMS in the US costs about $0.008 per message. Inbound voice is $0.0085/minute. A phone number is $1.15/month. My total messaging spend across 3 businesses sits between $5 and $15 a month — replacing tools I used to pay $150+/mo for.

STEP 1

Install the Twilio plug-in

Open Claude Code in your terminal. Inside Claude Code there is a plug-in marketplace most users never open. The Twilio plug-in adds dozens of expert-level skills — one for sending messages, one for voice calls, one for A2P compliance, one for AI voice agents, one for call recording, and many more — plus direct API access to your Twilio account. Once it is in, Claude can build, send, and operate everything for you in plain English.

PASTE THIS INTO CLAUDE CODE

```
I want to install the Twilio plug-in inside Claude Code so I can send texts, make voice calls, register A2P 10DLC, and build AI voice agents — all from plain English prompts.

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

Step 2: Find the plug-in named "twilio-developer-kit" published by the Twilio team. Install it.

Step 3: After it installs, type /twilio and confirm the autocomplete shows skills like /twilio-send-message, /twilio-voice-outbound-calls, /twilio-voice-conversation-relay, /twilio-call-recordings, /twilio-conversation-intelligence, /twilio-compliance-onboarding, /twilio-whatsapp-send-message, /twilio-rcs-messaging, /twilio-lookup-phone-intelligence, /twilio-taskrouter-routing, and several more.

Step 4: Confirm it is enabled by running /plugin again and searching "twilio" — you should see the plug-in marked enabled.

Confirm the install with me before moving on.
```

WHY THE PLUG-IN MATTERS

You can use Twilio without the plug-in. But every time you do anything, you would have to give Claude a tutorial on the API. The plug-in pre-loads expert-level knowledge of every part of Twilio's platform — Messaging, Voice, Verify, Conversations, ConversationRelay, TaskRouter, Conversation Intelligence — so Claude can build, deploy, and debug in one prompt instead of ten. It is the difference between hiring a senior messaging developer who already knows your stack vs. one who has to be onboarded every conversation.

STEP 2

Connect Claude to your Twilio account

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

PASTE THIS INTO CLAUDE CODE

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

Walk me through:
1. Where in the Twilio Console to find my Account SID + Auth Token
(and explain the difference — I want to use a Standard API key for
production, not the master Auth Token).
2. Creating a Standard API Key SID + Secret for this Claude Code
project.
3. Storing those credentials in a .env file in my project (and adding
.env to .gitignore so it does not commit).
4. Running a sanity check: list every phone number on my Twilio
account and show me my current account balance.

If this is a brand-new Twilio account with no numbers yet, just
confirm the connection is live and the balance is the trial credit.
```

API KEY VS AUTH TOKEN

Twilio gives every account a master Auth Token. Do not use it for anything you build. Create a Standard API Key for each project — it can be rotated, scoped, and revoked without taking down your whole account. If a key ever leaks, you revoke that key and the rest of your business keeps running. Claude will walk you through this in the prompt above.

STEP 3

Register A2P 10DLC (the step nobody warns you about)

If you are sending text messages to US customers from a business, the US carriers require you to register your brand + campaign before they let your texts deliver. This is called A2P 10DLC. Skip this step and your messages will silently get filtered as spam — customers will say "I never got your text" and you will think your stack is broken. It is not. You are unregistered. This is the single most common reason new business SMS programs fail to deliver, and almost no SaaS wrapper tells you about it until after you have signed up.

PASTE THIS INTO CLAUDE CODE

```
I need to register my business for A2P 10DLC so my SMS messages
actually deliver to US customers.
```

```
Walk me through the full registration:
1. Register my brand (legal business name, EIN or DUNS, address,
contact email, contact phone, business type, vertical).
2. Register my campaign (use case: customer care + marketing mix,
sample messages I plan to send, opt-in language I use on my website
forms).
3. Submit for carrier review.
4. Tell me what the approval timeline looks like (typical 1-3 weeks)
and what I can keep doing while I wait.

Use the /twilio-compliance-onboarding skill — it has the full A2P
10DLC playbook. Ask me each question in plain English, then file the
registration through the Twilio API.

Do not let me submit until you have flagged any field that is likely
to cause rejection (e.g., sample message that does not include an opt-
out, brand description that mismatches the registered business type).
```

MY A2P STORY

Summit Wraps got A2P-approved on May 10, 2026 — same day I shipped my first reactivation SMS campaign. 10 of 10 messages delivered. Gainz Train (my meal prep company) got initially rejected — the sample message I submitted did not include an opt-out phrase. Claude flagged it, I edited and resubmitted, approved on the second pass. The plug-in saves you from the most common rejection patterns. Without it, you submit blind and get rejected silently. Worth the 10 minutes.

STEP 4

Send your first SMS from one prompt

This is the moment Twilio stops being abstract and becomes infrastructure. You ask Claude to send a text. It sends it. You get a delivery confirmation. From that point on, every message your business needs to send — order confirmation, appointment reminder, reactivation blast — is one prompt away.

PASTE THIS INTO CLAUDE CODE

```
I want to send my first SMS through Twilio. Use the /twilio-send-
message skill.

Steps:
1. Use my A2P-registered Twilio phone number as the FROM number (look
```

```
it up from my account).
2. Send a test message to my personal cell: "[YOUR FIRST NAME], this is the first text from my business stack — built in Claude Code. Reply STOP to opt out."
3. Confirm delivery using the message SID + status callback. Show me the response.
4. If the delivery fails for any reason, diagnose it and tell me which carrier-side check failed and how to fix.
```

PASTE THIS INTO CLAUDE CODE

```
Now build me a reusable Python script (or Node.js — your call) that takes a CSV of contacts and a message template with merge fields (e.g. {first_name}, {last_appointment_date}) and sends a personalized SMS to each row.

Requirements:
1. Reads from .env for the Twilio credentials.
2. Throttles to 30 messages per second (Twilio's default per-number throughput) so we never hit a rate limit.
3. Logs every send to a local JSON file: timestamp, contact, message body, message SID, delivery status.
4. Has a --dry-run flag that prints what it WOULD send without actually sending.
5. Has a --confirm flag that requires me to type "YES" before bulk-sending to more than 10 contacts.

Test it on a 3-row CSV with just my own number. Show me the log file.
```

THE UNLOCK

The first time I ran that bulk send script, I sent 10 reactivation SMS to dormant Summit Wraps leads — personalized with each person's name and the vehicle they had quoted. Cost: 8 cents total. Click-throughs in the next 24 hours: 3. Time to build the script: about 12 minutes inside Claude Code. The same script — same code, same plug-in — now powers reactivation campaigns across every business I run.

STEP 5

Add a phone number for voice + voicemail

Twilio is not just SMS. The same number you provisioned can take inbound voice calls, route them, record them, and transcribe them. This is how RingCentral, OpenPhone, and CallRail all work — they are reselling Twilio numbers with a $25/mo wrapper around them. Once you own the underlying number, you can build that same wrapper yourself in plain English.

PASTE THIS INTO CLAUDE CODE

```
I want to add voice capabilities to my Twilio number so customers can call my business line.

Build me:
1. A simple inbound-call handler that greets the caller, then either (a) forwards the call to my personal cell during business hours (M-F 9am-5pm Mountain) or (b) sends them to voicemail outside those hours.
2. The voicemail flow: a recorded greeting, a 60-second message limit, automatic transcription, and an email to me with the audio attached + the transcript pasted inline.
3. A status callback that logs every call (inbound number, duration, recording URL, transcription) to a small JSON file I can audit weekly.

Use the /twilio-voice-twiml skill for the call flow and the /twilio-call-recordings skill for the voicemail capture. Deploy the call handler as a Cloudflare Pages Function from System 9 (so it runs 24/7 without my laptop). Wire the Twilio number's voice webhook to point at it.

Test by calling the number from my cell during business hours, then calling again after hours and leaving a voicemail.
```

REPLACING RINGCENTRAL

This single prompt replaces a $25-30/mo business phone subscription. The number costs you $1.15/month. Inbound minutes cost less than a penny each. Voicemail recording + transcription is bundled. For most service businesses doing fewer than 200 inbound calls a month, total monthly cost lands between $3 and $8. The setup time inside Claude Code is about 15 minutes — and unlike RingCentral, you own the data, you own the recordings, and you can pipe any of it into the rest of your stack.

STEP 6

Build an AI voice agent that answers your phone

This is the boss-level move. Twilio's ConversationRelay is a product that lets you build a real-time AI voice agent in front of any Twilio number. Calls come in, ConversationRelay handles the speech-to-text on the way in, your LLM (Claude, GPT, whatever) decides what to say, ConversationRelay handles the text-to-speech on the way out. The caller hears a natural conversation. The AI books appointments, answers FAQs, qualifies leads, or escalates to you when needed. All from one Claude Code prompt.

PASTE THIS INTO CLAUDE CODE

```
I want to build an AI voice agent that answers my business phone when
I am not available. Use the /twilio-voice-conversation-relay skill.

The agent should:
1. Answer with a natural greeting (use my actual voice via ElevenLabs
if I have a voice clone, or a clean default voice if not).
2. Detect what the caller wants: quote request, appointment booking,
support, or "just calling to talk to Brycen."
3. For quote requests: ask 3-4 qualifying questions (service type,
vehicle / location / timeframe), then promise a follow-up text within
1 hour and trigger a System 7 GHL workflow that posts the lead to my
Slack with full context.
4. For appointment bookings: check my Calendly or GHL calendar, offer
the next 3 available slots, book one if the caller confirms, send the
confirmation SMS through Twilio.
5. For "talk to Brycen": offer to take a detailed voicemail OR
transfer the call directly to my cell if it is during business hours.
6. For anything outside those four buckets: take a voicemail.

Deploy the ConversationRelay backend as a Cloudflare Worker from
System 9 (so it runs 24/7). Wire the Twilio number's voice webhook to
point at it. Use Claude Sonnet 4.6 as the LLM (cheap + fast). Add a
fallback path: if the agent gets stuck or the caller asks for a human,
transfer to my cell immediately.

Test by calling the number twice — once for a quote, once asking for
me by name.
```

EARN THE AUTO-PILOT

Same rule as the AI agents from System 8: default the voice agent to "draft mode" for the first 30 days. Have it transcribe every call + post the transcript to Slack, but do not let it actually book or quote yet. Watch what it would have done. Once you have 30 days of solid drafts, flip the switch to let it auto-book the narrow flows you trust (FAQ + voicemail capture are usually first). Save quote-giving and discount negotiations for human-in-the-loop forever, by design.

STEP 7

Record + transcribe + score every call automatically

Once your phone line is live, the next unlock is intelligence. Every call — inbound, outbound, voicemail — gets recorded, transcribed, scored against a rubric you define, and dropped into your CRM as a structured note. This is what Gong and CallRail charge $99-200 a month for. Twilio's Conversation Intelligence product does the same thing on the same per-minute usage pricing as your existing calls.

PASTE THIS INTO CLAUDE CODE

```
I want to add post-call intelligence to every voice call my business takes. Use the /twilio-conversation-intelligence skill.

For every completed call, automatically:
1. Transcribe the full conversation (already happens from Step 5 — confirm it).
2. Run a Conversation Intelligence pipeline that scores the call against 4 operators:
   - Intent: quote / booking / support / other
   - Outcome: booked / quoted / left voicemail / dropped
   - Customer sentiment: positive / neutral / frustrated
   - Did the agent (AI or human) ask for the sale: yes / no
3. Push the full transcript + the 4 scores into the contact's record in GHL (using the System 7 bridge) as a structured note titled "Call summary {date} — {outcome}".
4. If sentiment is "frustrated" OR outcome is "dropped," post an immediate alert to my Slack #ops channel with the transcript link.

Deploy this as a webhook on Twilio's recording-complete event. Show me a sample of what one analyzed call looks like inside GHL.
```

WHY THIS IS THE UNLOCK

Most service business owners have no idea what is being said on their own phone calls. Their team takes the call, takes a few notes, the rest evaporates. With this step, every call is permanent business intelligence. You can ask Claude "show me every frustrated call from the last 30 days and what they had in common." Or "show me every quote call that did not close — what objection came up most often?" Suddenly every conversation feeds back into the system. The pattern recognition compounds.

STARTER PROMPT LIBRARY

Three more dispatches to run today

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

PASTE THIS INTO CLAUDE CODE

```
Build me a scheduled Worker (on Cloudflare from System 9) that runs every weekday at 8am Mountain Time. For each contact in GHL with the tag "needs-quote-follow-up" whose last touch was more than 48 hours ago, send a personalized SMS through Twilio using a template I will provide. Log every send to a JSON file. Throttle to 30 messages per second. Stop sending the moment any single message gets a STOP reply.
```

PASTE THIS INTO CLAUDE CODE

```
Build me a phone-number fraud-check workflow that runs every time a new lead comes into my GHL. Use the /twilio-lookup-phone-intelligence skill to check: is this a real mobile number, is it a VoIP that is commonly used for scams, has it been flagged for SMS pumping in the last 30 days. If any flag fires, tag the lead "suspicious-phone" in GHL and skip auto-outreach. If clean, proceed with normal follow-up.
```

PASTE THIS INTO CLAUDE CODE

```
Build me a customer-onboarding text sequence for every new customer who pays an invoice in my GHL. Day 0: "Welcome — here's what happens next." Day 3: "Just checking in — any questions?" Day 7: "Quick favor — can you leave us a Google review?" Day 30: "Anniversary — here's a referral code." All four messages personalized with their first name + the service they paid for. Build it as a single scheduled Worker, NOT 4 separate workflows in GHL. Show me how to add new days to the sequence later in one line of code.
```

REAL RESULT

I run the messaging layer for 3 businesses on one Twilio account. Summit Wraps texts customers about install scheduling, sends reactivation campaigns, and receives inbound quote calls. Gainz Train sends order confirmations every Saturday at the cutoff. My personal brand uses it for SMS replies at scale. Total monthly Twilio bill across all 3: under $15. Before Twilio I was paying $30/mo for the GHL SMS add-on per business,

$25/mo for an OpenPhone line, and renting a ManyChat seat that I barely used — close to $150/mo of duplicative tools doing one job each. Now it is one plug-in inside Claude Code. A2P-registered, voice-ready, AI-agent-capable. Built every piece of this from a couch with zero coding experience.

That's the Whole System.

Ten systems. Zero ad spend. Zero coding experience. This is the exact stack that took my business from invisible to ranked on page one of Google, recommended by AI, auto-cross-posted to YouTube, built on a 40-page website that cost nothing, every lead followed up with before breakfast, the whole CRM operated in plain English, a 5-chief AI executive team running it all in parallel, the whole thing hosted on free Cloudflare infrastructure that runs 24/7 whether my laptop is open or not, and every text, voice call, and AI receptionist running through one messaging layer at pennies per message.

✓ **System 1: llms.txt**

AI chatbots now know what your business does, where you are, and who you serve. They recommend you by name.

✓ **System 2: Markdown Mirrors**

AI can read every page on your site cleanly. No HTML wrestling, no missed content, no hallucinated details.

✓ **System 3: Sitemaps + GSC**

Google knows exactly which pages matter. You can see your actual rankings, keywords, and where the quick wins are.

✓ **System 4: YouTube Autopilot**

Every video auto-cross-posts to YouTube with AI-optimized metadata. Search-friendly titles, keyword-rich descriptions, zero manual work.

✓ **System 5: Full Website Builder**

A complete 40-page website with SEO, schema, blog posts, and AI visibility. One prompt. One afternoon. Zero dollars.

✓ **System 6: CRM Lead Recovery**

AI scans your CRM every morning, finds missed leads, and drafts personalized follow-ups before you wake up. Never lose a lead again.

✓ **System 7: Claude Code + GHL**

Your CRM is now operated in plain English. Pull lead lists, draft personalized follow-ups, audit broken automations, build workflows. The operating layer for everything else in this playbook.

✓ **System 8: AI Executive Team**

5 chief agents + specialists, each with a charter, dispatched in parallel from one prompt. The system that runs the other seven without you holding it all in your head.

✓ **System 9: The Free Infrastructure Layer**

Cloudflare hosts every website, file, customer portal, scheduled task, and AI agent in your business. Free at the scale most service owners operate at. Replaces $180-500/mo in SaaS tools with one plug-in.

✓ **System 10: The Messaging Layer**

Twilio runs every text, voice call, voicemail, and AI receptionist across your business through one plug-in. A2P-compliant, voice-ready, AI-agent-capable. Pennies per message. Replaces $150-330/mo of duplicative messaging tools.

Want help implementing this?

I run a community where I share the ready-made scripts, answer questions, and do live builds. Post your site, get feedback, watch me build these systems for real businesses in real time. First 200 founding members lock in $97/mo for life.

Join the Community — $97/mo Founding Rate

Want me to build this for your business?

I'm taking on a small number of custom builds where I interview your business and build the entire system for you. Same tools, same stack — done in 2 weeks.

DM @brycenwood.ai

No coding experience was used in the making of this playbook. Or the systems it describes.

Built by Brycen Wood

IG: @brycenwood.ai | **TikTok: @brycenwood1** | **Web: summitwrapsandgraphics.com**

I document everything I build with Claude Code as a business owner with zero coding experience. Follow along for more systems, tools, and mistakes.

Community: brycenwood.com/community

If you build any of these, post it in the community or DM me. I want to see them work for you.

No coding experience was used in the making of this guide. Or the systems it describes.