Growth Protocol

Websites with Claude Code, part 1 of 3

The First Website *Playbook*

From a domain name to a live site with Claude Code, Astro and Cloudflare, in a few spare hours. No developer, no page builder, a Claude subscription and a domain. Every prompt ready to paste.

For founders, job seekers and solo marketers shipping a site for the first time.

By David Pires
growthprotocol.co

What is inside

A few hours, seven *moves*

Write the foundation in one sitting, build and polish in the next, publish in the third. After that, every change is a sentence.

Before any page

0 **Step 0: get comfortable**
What you will do that you have not done before · p.7

1 **The foundation**
Four short files: brand, copy, inspiration, SEO · p.7–9

2 **Step 1: install and make the folder**
Twenty minutes, once. Which model to use · p.10

3 **Step 2: first build**
One prompt, one private preview link · p.11

Ship it

3 **Step 3: preview and iterate**
Three rules for prompts, then CLAUDE.md · p.12–13

4 **Step 4: GitHub**
Your undo button and your handover · p.14

5 **Step 5: live on your domain**
Cloudflare, domain, analytics, Search Console · p.15

6 **Step 6: a way to reach you**
Booking link, email capture, a gated download · p.16

Then: the everyday loop (p.17), SEO rules (p.18), launch checklist (p.19), costs and tools (p.20), troubleshooting (p.21).

How to use this *playbook*

Every dark block is a prompt. Paste it into Claude Code as it is, then replace anything in angle brackets. Read what comes back before you move on. You will not write code. You will write four short text files and make decisions.

Before you start

Who this is *for*

Founders

Before product-market fit. You need a site that looks like a company, not a template, and you will rewrite it four times this year.

Job seekers and fractional leaders

A personal site with a point of view, a few proof points and a way to book a call. More than a PDF resume.

Solo marketers and side projects

Tired of waiting for someone else to change a headline. You want the site in your hands, with a way back when something breaks.

Not yet for you: a catalogue, user accounts, or five editors publishing daily. That is part 3 of this series.

What you will *have* at the end

- [ ] A live site on your own domain, with HTTPS
- [ ] Pages that load in under a second, on any phone
- [ ] Title, description, canonical and sitemap on every page
- [ ] A booking link and something that stores an email
- [ ] Visitor numbers, without a cookie banner
- [ ] Every version of the site saved, with a note on what changed
- [ ] A folder anyone can hand over or pick up
- [ ] Four text files that describe your brand, copy, taste and SEO rules
- [ ] A way to change any word on any page in one sentence
- [ ] A bill that is a Claude subscription and a domain

A few hours

spread over two or three sittings, if your copy exists

~$250

a year, and most of it is a subscription you may already pay

0 lines

of code you need to write or read

Before you start

What really made me *do* it

Not the technology. Three things I could not get from any platform I had paid for in fifteen years of Drupal, WordPress and Webflow.

01
It is free

Beyond a Claude subscription, every piece is free or open source: the framework, the code hosting, the site hosting, the analytics, the forms. A page builder starts cheap and adds a CMS plan, a language, storage, a form tool, an analytics add-on.

02
Zero dependency

No CMS to learn, no developer to wait for, no designer to brief. The design is written once in a text file and applied every time. Files in a folder, on a domain you control, built by tools you can read. One sentence takes you back a version.

03
Shipping with AI is a skill

Marketers, founders, job seekers: soon "I can ship with AI" will be on every CV that gets read. A website is the safest place to learn it. The result is public, the stakes are low, and you see every change.

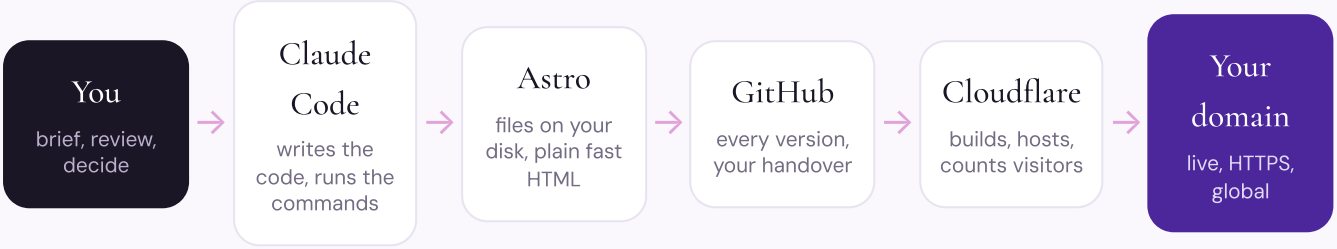
Rented versus *owned*

	Page builder	Claude Code + Astro
A new page	Drag blocks by hand, per page, per language	**A brief, then a preview**
A wording change on every page	Open every page	**One sentence**
Design consistency	Whatever the last editor did	**Written once in brand.md, applied every time**
SEO rules	In someone's head	**In seo.md, checked on every build**
Undo	Backups, if your plan has them	**Every version, forever, on GitHub**
Handover	Transfer the account, hope they know the tool	**Share a folder and a CLAUDE.md**
Cost	A plan, then add-ons, then help	**A Claude subscription and a domain**

Before you start

The stack, in one *picture*

Five pieces. You install a few things once, sign up for two free accounts, buy one domain. Every change, forever, follows the same path left to right.



Why *Astro*

A marketing site should be HTML, not an app. Astro ships pages with almost no JavaScript, so they are fast by default and Google reads them without effort. Free, open source, and any developer can read the files.

Why *Cloudflare*

Free hosting on a global network, HTTPS included, the domain at cost in the same dashboard, and cookieless visitor analytics with one toggle. Netlify and Vercel are fine too; Cloudflare keeps it in one place.

Why *GitHub*

Not a developer thing. It is the memory of your site: every version, with a note on what changed, and the way back when something breaks. It is also the handover: anyone can pick up the folder after you.

Why *Claude Code*

It runs in your folder, not in a chat window. So it can install, build, preview, test and fix, and it reads your foundation files before every change. A chat assistant can only hand you code to paste.

You install

Node.js, Git and Claude Code on your laptop. Twenty minutes, once. Page 10.

You sign up for

GitHub and Cloudflare. Both free, both take two minutes, both outlive any page builder.

You buy

The domain, at cost, at Cloudflare. And a Claude Pro subscription if you do not have one.

Not in the stack, on purpose: a CMS. For a one-to-ten-page site, the text lives in one file that Claude Code edits for you. Add an editing screen only when a second person needs to publish without you. That is part 3.

Before you start

Five moves. You own the first and the *last*.

Every change, from a typo to a new page, is the same loop. The judgement stays with you. Claude Code does the assembly in between.

1 · You

Brief

Goal, reader, angle, the facts and links it needs.

2 · Claude Code

Compose

Writes the page from your four foundation files.

3 · Claude Code

Preview

A private link, screenshots at phone and desktop width.

4 · You

Review

Facts, tone, links, every claim. Then ask for changes.

5 · Claude Code

Ship

Commit, push. Cloudflare publishes in a minute.

What a brief *looks* like

Paste into Claude Code

Add a page at /services/gtm-sprint. Goal: book calls from founders who just raised a seed round. Angle: three weeks to a message that sells, fixed price. Use the sprint description from copy.md, the two client quotes, and the booking link. One H1, search phrase "go-to-market sprint for B2B startups". Show me a preview at phone and desktop width before you commit.

What review *means*

- ◯ Every number and name is one you gave it
- ◯ No quote, logo or claim you did not supply
- ◯ Tone matches the three sentences in brand.md
- ◯ Links open the right place, in the right tab
- ◯ It reads well on your phone before your laptop

Claude Code is creative with words you did not give it. Review is the one step you cannot delegate.

Before any page

0 Get *comfortable*, then write the foundation

Thirty minutes in a Google Doc. No tools yet.

What to expect

You will do things you have not done on your computer. Open the terminal. Watch text scroll. Type `claude` and talk to a program that creates files. None of it requires understanding. It requires following the protocol. Claude Code asks permission before it runs a command; read the line, say yes. When something looks wrong, paste it into Claude Code and ask. That is the whole troubleshooting method.

Why a foundation *first*

Claude Code builds from what it can read. If all it can read is your prompt, you get a generic site. Give it who you are, what you say, what you admire and how Google should find you, and you get your site. Write four short sections in a Google Doc now. In Step 1 one prompt turns the doc into four files that every future change reads first.

`brand.md`

How you look and sound

- Positioning line
- Two colors
- Two fonts
- Words we use / never use
- Three sentences in my voice

`copy.md`

What the site says

- Headline
- For / not for
- Three offers
- Proof
- About, 80 words
- One call to action

`inspiration.md`

Three to five sites you like

- 1–2 for design
- 1–2 for content
- What each site is
- Take this, ignore that

`seo.md`

How Google reads it

- One phrase per page
- Title and description formula
- One H1
- Canonical
- Alt text
- Sitemap

Where to get the inputs: colors from your logo with any color picker; fonts from Google Fonts, one for headings, one for body; voice from a post or email you were proud of; proof from the three best things anyone said about your work; inspiration from the sites you keep coming back to.

brand.md and *inspiration.md*

brand.md

```
# Brand

## Who
<Name>. <One line: what you do, for whom.>

## Look
- Primary color: #<hex> (from the logo)
- Accent color: #<hex>
- Background white. Text near-black.
- Heading font: <Google Font>, weight 500
- Body font: <Google Font>, weight 400
- Logo: logo.svg in this folder. Never stretch it.
- Corners slightly rounded. No shadows. No
gradients on text.

## Voice
- First person. Short sentences. Plain words.
- Words we use: <five>
- Words we never use: <five>, no buzzwords, no
exclamation marks.
- Three sentences that sound like me:
  1. "<…>"  2. "<…>"  3. "<…>"

## Rules
- One idea per section. White space is allowed.
- Headings in sentence case. No all-caps labels.
- One primary button per screen.
- Mobile first: if it does not read at 390px, it is not
done.
```

inspiration.md

```
# Inspiration

Three to five sites. For each: the URL, what the
site is, what to take, what to ignore.

## For design (1 or 2)
- <url> — <a software product>. Take: the type
  scale, the calm spacing, how the hero breathes.
  Ignore: the dark theme.
- <url> — <a design studio>. Take: the footer,
  the way images sit in the grid. Ignore: the
animations.

## For content (1 or 2)
- <url> — <a solo consultant>. Take: the first
  sentence says who it is for and what changes.
  Ignore: the layout.
- <url> — <a B2B company>. Take: how proof
  sits right under the promise. Ignore: the jargon.

## Never
- Do not copy any of them. Borrow the principle,
  not the page.
```

Why the context matters: "what the site is" tells Claude Code why it is on the list, so it borrows the right thing and not the whole look. This file moves the result further than any other prompt.

copy.md and *seo.md*

copy.md

```
# Copy

## Hero
Headline: <8 to 12 words, the outcome>
Sub-headline: <one sentence, who it is for>
Call to action: <verb + noun> → <link>

## Who it is for
- <type of person, situation>
- <type of person, situation>
Not for: <one honest line>

## What I offer
1. <Name>: <one line>. <Format, duration, price if public.>
2. <Name>: <one line>.
3. <Name>: <one line>.

## Proof
Logos: <list>
Numbers: <two or three, with context>
Quotes:
- "<quote>" — <Name, role, company>
- "<quote>" — <Name, role, company>

## About
<80 words, first person: what you did, where, what you believe.>

## Final call to action
<One line> → <link>
```

seo.md

```
# SEO rules

## Every page
- One search phrase it answers, written at the top of the page file.
- Title: "<Page topic> | <Name>", under 60 characters.
- Description under 155 characters, written as the answer.
- Exactly one H1. H2s carry the sub-topics.
- A canonical URL, absolute, to itself.
- Open Graph title, description and image.
- Every image has alt text that says what is in it.

## Site
- sitemap.xml generated on build; robots.txt allows everything.
- www redirects to the bare domain, in one hop.
- No third-party script or cookie without asking me first.
- Build fails if a page has no title or no description.

## Phrases by page
- / : <phrase>
- /about : <phrase>
```

Do not have the copy yet? Write copy.md with Claude in a normal chat first, from your LinkedIn profile and three sentences about what you do. Then paste the result in. The site is only as good as this file.

Twenty minutes, once

I Install the tools and make the *folder*

The terminal is the app where you type commands. Mac: Cmd+Space, type Terminal, Enter. Windows: PowerShell from the Start menu. You use it for one thing: starting Claude Code.

1 **Node.js**

Download the LTS version from `nodejs.org` and install it like any app. It runs Astro on your machine.

2 **Git**

Mac: type `git --version` and press Enter; macOS offers to install it if missing. Windows: `git-scm.com`.

3 **GitHub**

Free account at `github.com`. Install the CLI from `cli.github.com`, type `gh auth login`, follow the questions. Once.

4 **Claude Code**

Install from `claude.com/claude-code`. Type `claude`, sign in with your Claude account.

Make the site *folder*

In Finder or Explorer: a folder inside Documents called `sites`, and inside it one named after your domain, for example `Documents/sites/growthprotocol`. Everything about the site lives there. Drop your logo in now. No logo? Skip it; Claude Code can make a wordmark.

Open Claude Code *there*

In Terminal, type `cd` and a space, drag the folder from Finder into the window (it pastes the path), press Enter. Type `claude`, Enter. Every time you come back: `cd`, drag, `claude`.

First prompt, with your foundation doc pasted after it

Here is my foundation doc. Split it into four files in this folder: brand.md, copy.md, inspiration.md and seo.md. Keep my words. Where something is missing, add a line that says TODO and what is needed.

Which *model*?

The default model on a Claude Pro plan is enough for everything in this playbook. If your plan offers a more capable model, switch to it with the `/model` command for the two moments where judgement matters most: the first build (Step 2) and the design iterations (Step 3). Switch back for everything mechanical: wording changes, commits, checks. Those run faster and cost less of your usage limit. Plan contents change; the Claude plans page says what each includes today.

Ten minutes

2 Build the *first* version

One prompt does most of the work. It reads your four files, visits the sites you like, and builds the whole thing with the technical basics in place.

Paste into Claude Code

Read brand.md, copy.md, inspiration.md and seo.md in this folder. Visit the sites in inspiration.md and note what I asked to take from each and what to ignore.

Set up a new Astro project here with static output and no UI framework. Build a one-page site that uses only the copy in copy.md, in this order: hero with headline, sub-headline and call to action; who it is for; what I offer; proof; about; final call to action; footer with email and LinkedIn.

Design: follow brand.md for colors, fonts and voice, and inspiration.md for layout, spacing and type scale. Self-host the fonts. Mobile first. Make it look like a site a designer made, not a template: generous white space, one accent color used sparingly, a clear type hierarchy, no stock illustrations.

Technical basics, done now: title, description, canonical and Open Graph tags from seo.md; one H1; alt text on every image; a sitemap and robots.txt; a favicon from logo.svg; a 1200 by 630 OG image with my name on the brand colors; a plain /privacy page with placeholder text I will replace. Do not add any analytics or third-party script.

Keep all text in src/data/site.ts so I can change words without touching components. Keep all styles in one global CSS file.

Then run the dev server and give me the local URL.

What *happens*

Claude Code asks one or two questions, asks permission to install and create things (say yes), creates about twenty files, and prints a link starting with `http://localhost:4321`. That link only works on your computer; it is your private preview. Open it. Your site exists. It will not be right yet. The first version is for seeing, not shipping.

What to *check* first

- [ ] Every word on the page is from copy.md
- [ ] The two colors are yours, nothing else
- [ ] Narrow the window to phone width: does it still read?
- [ ] All the words live in one file, site.ts. That is why later changes are a sentence.

An hour, maybe two

3 *Preview*, then iterate

The part only you can do: look at it on your laptop and your phone, and say what is wrong in plain words.

Three rules that save *hours*

1 **One change per prompt when it matters**
Four small prompts beat one long one. You see each result and know what caused what.

2 **Say what to keep**
"Keep everything else" stops Claude Code from redesigning the section you liked.

3 **Ask to see it**
Claude Code can screenshot the page at any width and compare it with your inspiration sites. Use it instead of guessing.

A typical iteration

The hero is too tall on mobile, cut the padding by a third. Show the logos in one row of five, grey, color on hover. Make the call-to-action button the primary color with white text. Keep everything else.

See it at two widths

Screenshot the full page at 390px and 1440px and show me both. Then compare the hero with the first site in inspiration.md and tell me the three biggest differences.

Change a word everywhere

Replace "consulting" with "sprints" everywhere on the site, then rebuild and show me where it changed.

Words that *work* in a prompt

Instead of	Say
"Make it better"	"Add 24px of space above each section heading"
"It looks off on mobile"	"At 390px the logos overlap. Stack them two per row."
"Make it pop"	"Make the headline 20% larger and put the accent word in italic in the primary color"
"Fix the colors"	"Only use the two colors in brand.md. Replace any other color with the nearest of the two."

End of Step 3

Write the note every future session reads *first*

A file called `CLAUDE.md` in the folder is read automatically at the start of every Claude Code session. It is how the site stays consistent in six months, and how someone else can take it over.

Paste into Claude Code

Write a CLAUDE.md for this project: what the site is and who it is for; where the text lives (src/data/site.ts) and the rule to edit words there only, never inside components; where styles live and the rule to add to the global CSS; brand.md in short form; the seo.md rules as a checklist; the rule that npm run build must pass before any change is done; the rule that no third-party script is added without asking me; and how to run, build and deploy. Under one page.

CLAUDE.md · what a good one looks like

```
# Claude Code notes for this site

- Text lives in src/data/site.ts. Edit words there, never inside components.
- Styles are global in src/styles/global.css. Add to the matching section, no scoped styles.
- Voice: first person, plain words, short sentences, no buzzwords. One italic accent word per heading.
- Brand: primary #<hex>, accent #<hex>, white background. Fonts: <heading> and <body>, self-hosted.
- SEO: one H1, title under 60, description under 155, canonical, OG image, alt text on every image. See seo.md.
- No third-party script or cookie without asking David first.
- After any change: npm run build must pass. Open the preview and check phone width.
- Deploy: commit and push to main. Cloudflare builds in about a minute.
```

This file is the handover. When you change roles or hire someone, they open the folder, read CLAUDE.md, run `claude`, and continue. No tour needed.

Thirty seconds

4 Push it to *GitHub*

Not a developer thing. GitHub is the memory of your site.

Paste into Claude Code

Initialize git in this folder, add a .gitignore for Node and Astro (node_modules, dist, .env), commit everything as "First version", and create a private GitHub repository named after this folder using the gh CLI. Push and give me the repository URL.

What you *get*

- Every version of the site, forever, with a note on what changed
- A way back: "undo the last change" works
- A backup that is not on your laptop
- The thing Cloudflare watches to publish your site
- The handover: share the repo, someone else continues

The habit from now *on*

After every change you like, one sentence:

After each change

Commit and push with a message that says what changed.

When something broke and you do not know why:

Undo

Show me the last five commits with what each changed. Then revert to the one before the hero change.

Words you will hear

Repository (repo): the folder, on GitHub. **Commit:** a saved version with a note. **Push:** send your commits to GitHub. **Main:** the branch Cloudflare publishes from. **Pull request:** a proposed change someone reviews first. You do not need the last one until a second person edits the site.

Fifteen minutes

5 Live on your *domain*

The only part that happens in a browser instead of a prompt.

Publish the *site*

1. Sign up at `cloudflare.com`. Free plan.

2. Create a new project from your GitHub repository (today under **Workers & Pages → Create → Connect to Git**; menu names move, look for "connect to Git"). Preset **Astro**, build command `npm run build`, output `dist`. Deploy.

3. A minute later you get an address ending in `.pages.dev`. Open it. Your site is on the internet.

4. **Analytics.** In the same project, turn on **Web Analytics**. Cookieless, so no banner. If you later add Google Analytics, HubSpot tracking or ad pixels, you will need consent; part 2 covers that.

Connect the *domain*

1. **No domain yet?** Cloudflare → Domain Registration. They sell at cost, no markup.

2. **Domain elsewhere?** Cloudflare → Add a site → enter the domain. It shows two nameservers; paste them at your registrar.

3. In the project: **Custom domains** (or a Domains tab) → add `yourdomain.com`, then `www.yourdomain.com`.

4. Cloudflare creates the DNS records and the certificate. Ten minutes later: live, HTTPS, free.

Back in Claude Code, once the domain works

Set the site URL in astro.config.mjs to https://<yourdomain.com> so the sitemap, canonical and Open Graph URLs use it. Redirect www to the bare domain in one hop. Make sure robots.txt points to the sitemap. Build, commit and push.

Then: add the domain in Google Search Console (free) and submit `/sitemap-index.xml`. Five minutes, and the moment Google starts counting.

6

Twenty minutes

A way to *reach* you

A site with no way to contact you is a brochure. Two things: a booking link and something that stores an email.

Booking *link*

Create a free one with HubSpot Meetings, Cal.com or Calendly. Copy the link.

Email *capture*

Anything that stores an address: a plain mailto link on day one, a Typeform or Tally form, or a free CRM like HubSpot if you want the contacts somewhere you can grow into. Create the form, copy the embed code.

Paste into Claude Code

Store the booking link https://<your-booking-link> once in site.ts and make every call-to-action button open it in a new tab. Add a contact section before the footer with this form embed: <paste the embed code>. Show a thank-you message on submit. If I did not give you an embed, use a mailto link to my email instead. Tell me if the embed sets any cookie before the visitor interacts with it.

A gated *download*

A checklist, a canvas, a guide, like this playbook. One page with the pitch and the form, one unlisted page the form redirects to with the file. Set the form's "redirect after submit" to the download page.

Paste into Claude Code

Add gated resources. Create src/data/resources.ts with one entry: slug "<name>", title, a two-sentence description, three bullets on what is inside, the PDF path /resources/<name>.pdf, and the form embed code. Build two pages per entry: /resource/<slug> with the pitch and the form; and /resource/<slug>/download, marked noindex and excluded from the sitemap, with a thank-you line and a download button. Put the PDF in public/resources/. Add a Resources section on the home page listing every entry.

Say what happens after the email, on the page. One follow-up note from you beats a five-email sequence for a first site.

From now on

The *everyday* loop

Open Terminal, `cd`, drag the folder, `claude`. Then one of these. A new page takes fifteen minutes; a wording change across the site takes one sentence.

A new page

Add a page at /<path>. Goal: <what it should make the reader do>. Reader: <who>. Search phrase: "<phrase>". Use this copy: <paste or point to a file>. Follow brand.md, inspiration.md and seo.md. Add it to the navigation and the sitemap. Show me a preview at phone and desktop width.

A new proof point

Add this quote to the proof section, after the second one: "<quote>" from <Name, role, company>. Keep the others.

A wording change everywhere

Replace "<old>" with "<new>" everywhere, including titles and descriptions. Show me the list of files that changed, then build and push.

A monthly check

Run the build, then check every page for: one H1, title under 60 characters, description under 155, canonical present, every image with alt text, no link that returns 404. Give me a table of what fails and fix the ones that are mechanical.

Speed check

Run Lighthouse on the home page at mobile settings and tell me the four scores and the two biggest things to fix. Fix them if they are in our code.

When you are stuck

Explain what you just changed and why, in plain words, in five lines. Then tell me what you would check first if the page looks wrong.

Connect your data, give the page a *goal*

The step up from here: connect Claude Code to Search Console or your analytics and brief pages from real numbers. "Which queries do we show for but rank below 10? Propose the page that would answer the top one." The page gets built from evidence, not from a competitor's menu. That is part 2.

Reference

SEO rules a first site should *never* break

Put these in seo.md and CLAUDE.md. Claude Code follows rules it can read; it cannot follow rules in your head.

- **One search phrase per page,** written at the top of the page file, and the page actually answers it
- **Title under 60 characters,** topic first, name last, one formula for every page
- **Description under 155 characters,** written as the answer, not as a teaser
- **Exactly one H1,** and the H2s carry the sub-topics a reader would scan
- **A canonical URL on every page,** absolute, pointing to itself
- **Open Graph title, description and a 1200 × 630 image** so shares look right
- **Alt text on every image** that says what is in it, not what it is for
- **A sitemap generated on every build,** submitted once in Search Console
- **www redirects to the bare domain** (or the reverse), in one hop, never both live
- **Every old URL redirects to its new page** in one hop, if there ever was an old site
- **No cookie before consent,** and no third-party script without asking
- **Once a month, the broken links Search Console reports get fixed.** One prompt.

Structured data worth adding on day *one*

Page	Schema	Why
Every page	Organization or Person	Tells Google and AI assistants who is behind the site, with the same name and description everywhere
Home, about	WebSite + BreadcrumbList	Clean site name in results, clean path in breadcrumbs
Resource pages	FAQPage	Your three or four questions can appear as rich results and are easy for assistants to quote
Articles, later	Article	Author, date, image: the fields assistants use to decide what to cite

Paste into Claude Code

Add JSON-LD structured data: Person (or Organization) on every page using the name, description and links from site.ts; WebSite and BreadcrumbList on the home page; FAQPage on any page that has a FAQ list. Validate against schema.org types and show me one example per type.

Reference

Launch *checklist*

Thirty minutes before you share the link. Tick every line, then post it.

Content

- [ ] Every word is from copy.md or written by you
- [ ] No placeholder text, no "lorem", no default image
- [ ] Every name, number and quote is real and approved
- [ ] About section has a photo of you
- [ ] Email and LinkedIn in the footer are correct

Technical

- [ ] `npm run build` passes with no warnings
- [ ] Site loads on https://yourdomain.com and www redirects
- [ ] `/sitemap-index.xml` and `/robots.txt` exist
- [ ] Lighthouse mobile: performance above 90
- [ ] Favicon shows in the tab

SEO and sharing

- [ ] Title and description on every page, in the formula
- [ ] One H1 per page, checked in page source
- [ ] OG image shows when you paste the link into LinkedIn
- [ ] Domain added to Google Search Console, sitemap submitted
- [ ] Structured data validated

Conversion and privacy

- [ ] Booking link opens and the calendar shows real slots
- [ ] Form submits and you received the test
- [ ] Gated download: form redirects, PDF opens
- [ ] Cloudflare Web Analytics is on; no other tracker loads
- [ ] Privacy page exists and the footer links to it

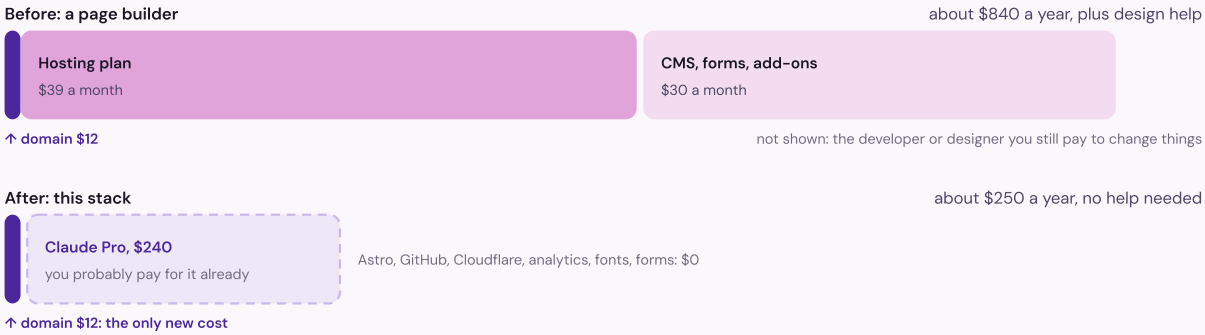
Let Claude Code run the technical half

Run the launch checks: build passes, every page has title, description, canonical, one H1 and alt text on all images, sitemap and robots.txt exist, no internal link returns 404, the www redirect works, no third-party script loads, and Lighthouse mobile performance is above 90 on the home page. Report each as pass or fail with the fix.

Reference · prices checked October 2026, typical mid-tier plans; this page is the only one with numbers

What it costs, and the *tools*

Before and *after*, per year



Three times cheaper, and most of what is left is a subscription you already have.

Prices as of October 2026, typical mid-tier plans. The point is not the saving: the cheaper option is also the one you control.

Tool	Purpose	Price	Free tier enough?
Claude Pro	Runs Claude Code	$20 a month	Yes, for one site
Astro	Website framework	Free, open source	Yes
Node.js	Runs Astro on your machine	Free	Yes
GitHub (with Git and the gh CLI)	Versions, backup, handover	Free	Yes
Cloudflare (Pages, Registrar, Web Analytics)	Hosting, domain at cost, cookieless analytics	Free; domain $10 to $15 a year	Yes
Google Fonts	Fonts, self-hosted	Free	Yes
Google Search Console	How Google indexes you	Free	Yes
HubSpot	Free CRM: forms, booking link, contacts	Free tier	Yes
Typeform or **Tally**	Forms without a CRM	Free tiers	Yes
Cal.com	Booking link alternative	Free tier	Yes
Canva	OG image, social visuals	Free	Yes

Reference

When it *breaks*

What you see	What to do
"command not found: claude" or "node"	The install did not finish or Terminal needs a restart. Close Terminal, open it again, try once more. Then re-run the installer.
Claude Code says it is not in the site folder	You skipped the cd step. Type cd and a space, drag the folder into Terminal, Enter, then claude.
The preview link shows a blank page or an error	Paste the error to Claude Code: "The dev server shows this error. Fix it and tell me what it was."
Cloudflare build fails	Open the build log, copy the last 20 lines into Claude Code. Usually a missing dependency or the wrong output folder (it must be dist).
The domain shows a Cloudflare error page	DNS still propagating (up to an hour) or the domain was not added to the project. Check Custom domains shows "Active".
The site looks wrong after a change	"Show me the last commits. Revert to the one before <change>, build, push." Then try the change again in smaller steps.
Fonts look like Arial	"Check that the fonts in brand.md are self-hosted in public/fonts and referenced in the CSS. Fix."
You hit the usage limit mid-build	Wait for it to reset, then continue; the files are saved. Next time, do mechanical changes on the cheaper model.

Built it? *Show me.*

Send the link to hello@growthprotocol.co. I read every one, and this playbook gets better because of yours. Twice a year I re-run Step 2 on a fresh folder; if it still ships a site, the playbook stands.

Part 1 of 3. This is the fast version.
Part 2: a bigger site that already exists, on Webflow or WordPress, moved without losing rankings.
Part 3: teams, rules, review, hundreds of pages.

growthprotocol.co · David Pires

Licensed CC BY 4.0. Use it, share it, credit it.