Launch your own website this weekend

Build and launch your own website this weekend, from an empty folder to your own domain, with an AI coding agent doing the coding. No coding experience needed.

by Taku

What it costs

Checked 2026-11-21 and 2026-11-22.

Piece	Price	Source
AI coding agent	Claude Pro $20 a month billed monthly (includes Claude Code). ChatGPT Plus $20 a month, which lists Codex in the command line	claude.com/pricing, ChatGPT Codex pricing
GitHub	Free plan $0, unlimited private repositories	github.com/pricing
Design skill	Free, open source	Emil Kowalski, Impeccable, Taste Skill
Supabase (only if you need logins or data)	Free plan $0. Free projects pause after 1 week with no activity, 2 active free projects	supabase.com/pricing
Vercel hosting	Hobby plan $0, for personal or non-commercial use only. A site for a business that earns money needs a paid plan	Vercel Hobby, Vercel Terms
A .com domain	About $10 the first year at Namecheap. Renewal after that: about $20 a year	Namecheap .com

So a personal site costs about $30 for your first month: a $20 agent subscription plus a domain at about $10 for the first year. After that, the subscription repeats monthly if you keep it, and the domain renews at about $20 a year.

How it works

The agent checks your computer and asks a few questions about your site (Stage 0), then takes you through seven stages, one at a time. You make the accounts and you pay for anything you buy; the agent does the code and tells you exactly what to click.

The agent is Claude Code or Codex. The tools for each stage:

0 **Orient yourself, then ask only what is missing**
Node.js, git, GitHub CLI

1 **The project and GitHub (a saved copy of every version)**
GitHub, Next.js

2 **A design skill, so it does not look like AI slop**
Emil Kowalski's skills, Impeccable, Taste Skill, skills installer

3 **The first version, on your own computer**
Next.js

4 **Supabase, only if you need logins or to store data**
Supabase, Supabase MCP

5 **Launch it on Vercel**
Vercel, Vercel and Git

6 **Your own domain**
Namecheap, GoDaddy, Vercel domains

7 **Launch checklist and how to keep going**
git revert, Vercel rollback

The rules the agent works by

- **Plain words.** Explain every step in one or two short sentences before you do it. No jargon without a one-line meaning ("a repository, the folder GitHub keeps for your code").
- **One question at a time,** with two or three example answers when a question is abstract.
- **You do the commands, they do the accounts.** Run terminal commands yourself and say what each one does. Creating accounts, signing in, entering payment details and buying a domain are always done by them, in their own browser. Never type a password, card number or one-time code, and never ask them to paste one into the chat.
- **Secrets stay out of the code.** API keys go in a `.env.local` file that is listed in `.gitignore` before the first commit, and into Vercel's environment variables. Never commit them, never print them back.
- **Nothing is deleted without asking.** Before you remove or overwrite a file they made, say which one and get a yes.
- **Save often.** After every change that works, commit it with a one-line message in plain words, and once Stage 1 is done, push it to GitHub. A commit on GitHub is what lets them roll back.
- **Keep your place.** Keep a short `PROGRESS.md` in the project: the stage you are on, their answers from Stage 0, the choices made. Update it at the end of each stage. If the session restarts, they say "read PROGRESS.md and carry on" and you pick up there.
- **Never invent facts about them** or their business on the site. If you do not know something (opening hours, prices, a quote from a client), leave a clearly marked placeholder and list it for them at the end.
- **Check before you say done.** Each stage ends with a check you can see (a command's output, the page in the browser). If something fails, read the error, fix it, and tell them in one line what happened.
- Match their language and tone.

Stage 0: Orient yourself, then ask only what is missing

Silently, before your first question, find out what you can:

- what is in this folder already (an existing project, images, a logo, text they wrote);
- which tools are installed: run `node -v`, `git --version`, `gh --version`;
- whether a `PROGRESS.md` already exists (then this is a restart: read it, say where you were, and carry on from there);
- anything they attached or said in this conversation.

Then say back in three to five lines what you found ("You have Node 26 and git, no GitHub CLI yet; the folder has a logo and a menu PDF"). Fix missing tools first, using the commands in the appendix: Node.js LTS, git (on Windows, Git for Windows), and the GitHub CLI. Installers that open a window or ask for the computer's password are run by them, not you; on Windows they reopen PowerShell afterwards. Use the current Node.js LTS: the skills installer needs Node.js 22.20 or later.

Then ask, one at a time, **only what you could not find**:

1. What is the site for, and who is it for? (a portfolio, a business, a newsletter, an event)
2. Is it personal, or for a business that earns money? (This decides the Vercel plan in Stage 5.)
3. Which pages do you want to start with? (Home only is a fine start; for example Home, About, Contact.)
4. Do visitors need to log in, or does the site need to store anything, like sign-ups or orders? (If no, Stage 4 is skipped.)
5. Do you have a name and a domain in mind? (for example `yourname.com`)
6. Two or three websites you like the look of, and what you like about them.
7. Any text, photos or logo you already have. Point me at them or drop them in this folder.

Sum up their answers in four lines and ask "anything to change?" before Stage 1.

Stage 1: The project and GitHub (a saved copy of every version)

GitHub is where developers store their code. It keeps every version, so if something breaks you can go back to one that worked.

1. **They make a free GitHub account** at github.com/signup (GitHub's steps), then they sign the GitHub CLI in: `gh auth login --web --git-protocol https` (gh auth login). It shows a one-time code to type into the browser page it opens, and waits until they finish, so have them run it themselves (in Claude Code they can type `! gh auth login --web --git-protocol https`), or run it in the background and wait for them to say done.

   Then check who the saves are from: `git config --global user.name`. If nothing is set, set `git config --global user.name "Their Name"` and `git config --global user.email "<their GitHub no-reply address>"` (GitHub Settings, Emails). If something is set, ask before changing it.

2. **Create the project** in this folder. Use Next.js unless they asked for something else: `npx create-next-app@latest . --yes` (the Next.js docs default: TypeScript, Tailwind CSS, ESLint, App Router). The setup refuses a folder with other files in it, so if their files are here (their logo, photos, this playbook), move them to a folder next to it, for example `../my-site-files`, run the command, then move images into `public/` and other files back into the project.
   - If Stage 4 will be needed (logins or data), use Supabase's starter instead: `npx create-next-app -e with-supabase .` (Supabase Next.js quickstart).

3. Check `.gitignore` has a line that covers `.env.local` (`.env*` or `.env*.local`). The setup makes the first commit itself; run `git log` to check, and commit now if there is none.

4. **Create a private repository and push:** `gh repo create <site-name> --private --source=. --push` (gh repo create).

Check: run `gh repo view --web` so they see their code on GitHub. Tell them: "Every time we save and push from now on, GitHub keeps that version."

Stage 2: A design skill, so it does not look like AI slop

Left alone, coding agents tend to reach for the same safe defaults. Impeccable's own README lists them: "Inter for everything. Purple-to-blue gradients. Cards nested in cards" (pbakaus/impeccable). A design skill gives the agent rules and taste to work from. Show them these three (all free and open source), with one line on each, and let them pick one:

Skill	What it is, in its own words	Install (run in the project folder)
Emil Kowalski's skills (repo)	"Skills for Designers and Engineers"; most of its skills are about animation and interface details, "based on my years of experience working at companies like Vercel and Linear"	`npx skills@latest add emilkowalski/skills`
Impeccable (impeccable.style, repo)	"The design vocabulary for your coding agent"; commands like `critique`, `polish` and `bolder`, plus a detector that flags AI slop	`npx impeccable install`, reload the agent, then `/impeccable init` (`$impeccable init` on Codex)
Taste Skill (tasteskill.dev, repo)	"Agent skills that give coding agents taste: art direction, reference images and a browser check"; built for company websites; needs Node 22+ and Chrome, Edge or Chromium	`npx skills add https://github.com/Leonxlnx/taste-skill --skill "design-taste-frontend"`

How to help them choose: Taste Skill if they want a full company page built with art direction; Impeccable if they want to steer and review the design step by step; Emil Kowalski's skills if motion and polish matter most. They can add a second one later.

`npx skills` is the open skills installer; it puts skills where Claude Code and Codex read them (where it installs). If an installer asks questions, read each one to them and let them answer. The skills installer may show a security rating table for a skill; read it to them, explain that a skill runs with the agent's permissions, and let them decide. Before you run any installer, open its README and confirm the command above is still the one it shows.

Check: the skill appears in your skills list (on Claude Code, restart the session if it does not). Add `.agents/**` and `.claude/**` to the ESLint ignores in `eslint.config.mjs`, so the skill's own files do not show as errors in your project. Then write a short `DESIGN.md` in the project (if they chose Impeccable, run `/impeccable document`, or `$impeccable document` on Codex, to write it instead) with the direction you agreed: the feel in their words, the sites they liked, two fonts, a colour palette, and what to avoid. Show it to them and get a yes.

Stage 3: The first version, on your own computer

1. Start the local preview with `npm run dev` in the background (it keeps running; stop it when you are done) and give them the address (usually `http://localhost:3000`) to open in their browser.
2. Build the pages from Stage 0 using the design skill and `DESIGN.md`. Use their real text and images. Placeholders are clearly marked.
3. Show them each page and ask for notes. Make the changes, then commit.
4. Before you move on, check the site on a phone-sized window, check every link works, and run `npm run build` to make sure it builds without errors.

Check: `npm run build` passes, they have seen every page and said it is ready for a first launch. It does not have to be perfect: they can keep changing it after it is live.

Stage 4: Supabase, only if you need logins or to store data

Skip this stage if the answer to question 4 was no.

Supabase gives the site a database and logins. The free plan covers 50,000 monthly active users and a 500 MB database; free projects pause after a week with no activity, and you can have 2 active free projects.

1. **They make a free account and a new project** at supabase.com.
2. **Connect yourself to it** so you can create and check the tables directly (Supabase MCP docs):
   - Claude Code: `claude mcp add --scope project --transport http supabase "https://mcp.supabase.com/mcp?features=docs%2Caccount%2Cdatabase%2Cdebugging%2Cdevelopment%2Cfunctions%2Cbranching%2Cfeedback%2Cnotebooks"`, then they type `/mcp`, select "supabase" and "Authenticate".
   - Codex: `codex mcp add supabase --url "https://mcp.supabase.com/mcp?project_ref=<project id>"`, then `codex mcp login supabase` (how Codex adds MCP servers).
   - Scope it to this one project, as the Supabase page advises: on Claude Code, add `&project_ref=<project id>` to the end of the address (the id is in Project Settings, General). This is a new, empty project, so leave write access on while you build; once real people's data is in it, re-add the server with `&read_only=true` (Supabase advises read-only mode for production).
3. Put the project URL and the publishable key in `.env.local` as `NEXT_PUBLIC_SUPABASE_URL` and `NEXT_PUBLIC_SUPABASE_PUBLISHABLE_KEY` (quickstart): get them through the Supabase connection, or have them copy both from the dashboard (Project Settings, API Keys) into `.env.local` themselves. The publishable key is made for the browser; the secret key never goes in the chat or in any file.
4. Build what they need (a sign-up form, a login page, a table for orders), with row level security on every table.

Check: sign up as a test user on the local site, see the row appear in Supabase, then commit.

Stage 5: Launch it on Vercel

Vercel puts the site on the internet. Once it is connected to GitHub, every push to the main branch goes live by itself (Vercel and Git), and Vercel says it deploys "in seconds" (pricing).

Plan: the Hobby plan is free, but its terms allow "personal or non-commercial use" only. If the answer to question 2 was a business that earns money, tell them plainly that they need Vercel's paid Pro plan, show them vercel.com/pricing, and let them decide.

1. **They sign up** at vercel.com with their GitHub account.
2. **They import the repository:** New Project, pick the repo, Deploy (import a project). If Stage 4 ran, they add the two Supabase variables under Environment Variables first (environment variables); a change to them only applies to the next deployment.
3. Open the `.vercel.app` address it gives them together.

Check: the live address loads every page. Then show them the loop: change one word, commit, push, and watch it go live.

Stage 6: Your own domain

1. **They buy the domain** at a registrar they choose: Namecheap or GoDaddy, or directly in Vercel, which offers "seamless integration with Vercel". Remind them the first-year price is not the renewal price (see the cost table).
2. In Vercel: the project's Settings, Domains, click Add Existing and type the domain; when Vercel offers to add `www` too, say yes (add a domain).
3. If they bought it elsewhere, Vercel shows the DNS records to set. Walk them through adding exactly those records in their registrar's DNS settings. **Copy the values Vercel shows,** do not use an address from memory: Vercel's own troubleshooting page says to "use the A record value shown in your project's Settings > Domains."
4. DNS can take a while to update. Check the Domains page until it shows the domain as valid.

Check: `https://<their-domain>` loads the site with no "Not secure" warning in the address bar.

Stage 7: Launch checklist and how to keep going

Go through this with them:

- Every page loads on a phone and a laptop, every link and button works.
- The page title and description are set for each page (what shows in Google and when the link is shared).
- No placeholder is left; list any that are, for them to fill.
- No key or password anywhere in the repository (search for it).
- The contact details, prices and claims on the site are theirs and true.

Then tell them how to keep going:

- **To change anything:** open the agent in this folder, say what you want, check it on `localhost`, and ask it to commit and push. Vercel says it deploys "in seconds".
- **If something breaks:** say "roll back the last change". The agent uses `git revert` (git revert), which adds a new save that undoes the bad one, and pushes it. On Hobby, Vercel's own instant rollback only goes back to the previous deployment (rollback), so `git revert` is the way back further.
- **To improve the design:** run the design skill's review on one page at a time (for example `/impeccable critique the home page`).

Finish with: "You have launched your own website." Then write a `NEXT-STEPS.md` in the project with the open placeholders, the accounts they made (names only, never passwords), and what renews when, and mark `PROGRESS.md` as done.

Install commands

Checked 2026-11-22. You run the installers yourself.

Tool	macOS	Windows	Source
Claude Code	`curl -fsSL https://claude.ai/install.sh | bash`	`irm https://claude.ai/install.ps1 | iex` (PowerShell)	Claude Code setup
Codex	`curl -fsSL https://chatgpt.com/codex/install.sh | sh` or `npm install -g @openai/codex`	`powershell -ExecutionPolicy ByPass -c "irm https://chatgpt.com/codex/install.ps1 | iex"`	openai/codex
Node.js (LTS)	installer from nodejs.org	Windows Installer (.msi) from nodejs.org	nodejs.org/download
git	`xcode-select --install` or `brew install git`	`winget install --id Git.Git -e --source winget`	git for macOS, git for Windows
GitHub CLI	`brew install gh` (needs Homebrew; without it, use the macOS installer on cli.github.com)	`winget install --id GitHub.cli`	GitHub CLI

Start the agent inside your site's folder: `cd my-site`, then `claude` or `codex`. The free claude.ai plan does not include Claude Code (setup).

How to use it

1. Get an AI coding agent: Claude Code (included in Claude Pro, $20 billed monthly) or Codex (ChatGPT Plus, $20 a month, lists Codex "in the CLI"). The install commands are in the appendix; you run the installers yourself, since some open a window or ask for your computer's password. On Windows, close and reopen PowerShell after installing anything.

2. Make an empty folder for your site, for example `my-site` on your desktop. Open Terminal (Mac) or PowerShell (Windows), type `cd` (with a space), drag the folder onto the window and press Enter. Then type `claude` or `codex`. The first time, it asks you to sign in to your Claude or ChatGPT account, and it asks before it runs each command: read the one-line reason and allow it.

3. Attach this file in the chat (drag it in, or paste it), and say: **let's build my website**.

4. If you stop and come back later, or the session restarts: open the agent in the same folder, attach this file again and say **read PROGRESS.md and carry on**.

> Open Claude Code or Codex in your site's folder, attach the playbook and say: **let's build my website**

The playbook and this PDF: velorum-ai.com/resources/website

After the launch

You don't need us for any of this. Where we help at Velorum is the next step for a business: turning the jobs your team repeats into agents and automations that run on their own, connected to the tools you already use.

Book a call with Velorum

Sources

Every claim in this guide, where it comes from and when it was checked. The playbook links each one again next to the line it supports.

1. Claude Pro $20 billed monthly, includes Claude Code; free plan does not (checked 2026-11-22)
   https://claude.com/pricing https://code.claude.com/docs/en/setup
2. ChatGPT Plus $20 a month, Codex "in the CLI" (checked 2026-11-22)
   https://learn.chatgpt.com/docs/pricing
3. GitHub Free $0, unlimited private repos (checked 2026-11-21)
   https://github.com/pricing
4. GitHub stores "each file's revision history" (checked 2026-11-21)
   https://docs.github.com/en/repositories/creating-and-managing-repositories/about-repositories
5. Impeccable README: "Inter for everything. Purple-to-blue gradients. Cards nested in cards" (checked 2026-11-22)
   https://github.com/pbakaus/impeccable
6. Emil Kowalski: "based on my years of experience working at companies like Vercel and Linear"; most skills on animation (10 to 12 of 19) (checked 2026-11-22)
   https://github.com/emilkowalski/skills
7. Taste Skill description (checked 2026-11-22)
   https://github.com/Leonxlnx/taste-skill
8. Supabase free: 50,000 MAU, 500 MB, pauses after 1 week, 2 active projects (checked 2026-11-22)
   https://supabase.com/pricing
9. Supabase MCP: AI tools "interact with and query your Supabase projects" (checked 2026-11-21)
   https://supabase.com/docs/guides/ai-tools/mcp
10. Vercel auto deploys on push to production branch (checked 2026-11-22)
    https://vercel.com/docs/git
11. Vercel "deploy in seconds" (checked 2026-11-21)
    https://vercel.com/pricing
12. Vercel Hobby free; "personal or non-commercial use" (checked 2026-11-22)
    https://vercel.com/docs/plans/hobby https://vercel.com/legal/terms
13. Namecheap .com $12.08 first year, $6.48 new customer, $19.98 renewal (checked 2026-11-22)
    https://www.namecheap.com/domains/registration/gtld/com/
14. GoDaddy .com $22.99 a year list (checked 2026-11-22)
    https://www.godaddy.com/resources/skills/how-much-domain-name-cost
15. Vercel A record: use the value shown in Settings > Domains (checked 2026-11-22)
    https://vercel.com/docs/domains/troubleshooting