CLAUDE-SEO · PROMPT PACK

Build, optimize & publish a website.

Run these in Claude Code one at a time. Where there's a **/command**, that's the one to run, just swap the **[brackets]** for your own. No command? Say the line under it. Both work, the command is faster. Click anything to copy it.

Built around **claude-seo**, the skill that sets up, optimizes, and ranks the site. Install it first. One step uses **claude-ads** for brand colors and voice, and it's optional.

In the video we use **claude-ads.md** as the example site. The same steps work for any product, service, or skill.

---

0 Before you start

- Install **claude-seo**.
- Pick your thing: a product, a service, a skill, a business. Keep the real info in one place (a repo, a doc, or a page).
- Grab a template you like from v0.dev and drop the folder into your project.
- Open Claude Code inside that folder.

---

1 Strategy and plan

1 Make the page plan

> Read [my repo or doc] and the template in this folder. Plan the pages this site needs. Show me the plan before you build.

2 **Get an SEO plan**

```
$ /seo plan [saas | local | ecommerce | agency | publisher]
```

OR SAY Make me an SEO plan for [my type of business].

3 **Find the keywords**

```
$ /seo cluster [my main topic]
```

OR SAY Find the keywords and topic groups for [my main topic].

4 **Set the audience**

› This site is for [who it's for]. Write every page for them. Lead with what they care about most.

---

2 Build the pages

5 **Keep the template**

› Keep this template's layout. We're swapping the placeholder text, images, and links for our own.

6 **Write the hero**

› Write the hero. One bold line about [main benefit], a clear subline, and two buttons.

7 **Build the homepage**

› Build the homepage from [my repo or doc]. Real info, real numbers, no hype. Match the template's layout.

8 Build the other pages

> › Now build the rest of the pages from the plan: [list your pages, like features, pricing, about, contact].

9 See it

> › Start the preview so I can see it.

---

3 Write the content

10 Brief each page

```
$ /seo content-brief [page topic]
```

> OR SAY Give me a brief for the [page] page: what to say and what to rank for.

11 Add an FAQ

> › Add an FAQ that answers the real questions [audience] asks before they buy.

12 Add proof

> › Add a section for proof: stars, numbers, reviews, logos, whatever is real.

13 Add a comparison page

```
$ /seo competitor-pages [url]
```

> OR SAY Add a page comparing us to [alternative]. Honest, clear, with a simple table.

14 Tighten the copy

> › Read every page and tighten the copy. Short, clear, benefit first. Cut the fluff.

---

4 Links and navigation

15 Wire the links

> › Add a menu, connect every page, and fix anything broken. Keep my outside links as buttons.

16 Add internal links

```
$ /seo cluster
```

> OR SAY Use the keyword groups to link related pages to each other.

---

5 Brand and visuals

17 Pull your brand (optional)

```
$ /ads dna [my site or repo]
```

> OR SAY Pull my brand: colors, fonts, and voice.

Optional, uses the claude-ads skill.

18 Brand pass

> › Match my brand: [your colors], [your vibe]. Keep it clean and simple.

19 Images and alt text

```
$ /seo images [url]
```

OR SAY Add a logo icon and a social preview image. Add a short description to every image.

20 Make an image

```
$ /seo image-gen og
```

OR SAY Make a social preview image for [page].

---

6 On-page SEO

21 Titles and descriptions

› Write a unique title and short description for every page. Clear, and built around the keyword.

22 Check a page

```
$ /seo page [url]
```

OR SAY Check the [page] for on-page SEO and fix what's weak.

23 Add structured data

```
$ /seo schema [url]
```

OR SAY Add the structured data for the site, the business, and the product. Put it in the page source.

---

7 Technical SEO

24 Technical check

```
$ /seo technical [url]
```

OR SAY Run a technical check and fix the crawl, speed, and mobile issues.

25 Site map and crawl files

```
$ /seo sitemap generate
```

OR SAY Make the site map and the crawler files. Let the AI search bots in too.

---

8 AI search (GEO)

26 Make it AI-search ready

```
$ /seo geo [url]
```

OR SAY Make it ready for AI search: a clear answer near the top, question-style headings, and short sections that are easy to quote.

9 Quality

27 Mobile, speed, access

> Check it on mobile and fix anything that breaks. Make it fast and easy to read.

28 Content quality

```
$ /seo content [url]
```

OR SAY Check the content for trust and depth, and flag any thin pages.

10 Audit and fix (during the build)

29 Full audit

```
$ /seo audit [url]
```

OR SAY Run a full SEO audit on the site.

30 Fix what it finds

> Fix everything it found, then run the audit again.

Repeat 29 and 30 until it comes back clean.

Phase A Final verification (before you host anything)

A1 Full audit

```
$ /seo audit [url]
```

OR SAY Run a full SEO and quality audit on the whole site. List everything that's wrong, grouped by priority, blockers first. Don't fix anything yet, just show me the report.

A2 Technical and crawl check

```
$ /seo technical [url]
```

OR SAY Run a technical check: crawlability, speed, mobile, broken links, redirects, and console errors. Tell me what would stop this from ranking or loading.

A3 Links and navigation

› Check every internal and external link on the site. Flag anything broken, any 404, any link pointing at localhost or a placeholder. Make sure the menu reaches every page.

A4 Meta, schema, and titles

```
$ /seo schema [url]
```

OR SAY Verify every page has a unique title and description built around its keyword, and that the structured data (site, business, product) is present in the page source. List any page that's missing them.

A5 Content and AI-search readiness

```
$ /seo content [url] then /seo geo [url]
```

OR SAY Check the content for depth and trust, flag any thin pages, and confirm each page has a clear answer near the top with question-style headings so AI search can quote it.

A6 Fix loop

› Fix everything the audit and checks found, then run the full audit again. Repeat until it comes back clean.

Run A6 as many times as it takes. Same idea as steps 29 and 30.

A7 Save a baseline

```
$ /seo drift baseline
```

OR SAY Save a baseline of the current state now, so we can catch any drops after launch.

A8 Production build

› Build the site for production and fix any build errors. Show me the output folder and confirm it builds with zero errors.

Don't move to Phase B until A8 is green.

Phase B Deploy to Railway (CLI)

Heads-up: **railway login** opens a browser, so Claude Code can't do that step for you. You

click through it yourself. Everything else it can run.

B1 Install and log in

```
$ npm i -g @railway/cli
```

OR SAY Install the Railway CLI globally if it isn't here already. Then give me the exact command to log in, and pause so I can authenticate in my browser. If browser login won't work here, use the browserless login flow and show me the pairing code and URL.

B2 Create the project and deploy

```
$ railway init then railway up
```

OR SAY I'm logged in. Run railway init to create a new project for this site, then railway up to deploy the production build. When it finishes, give me the temporary up.railway.app URL and confirm the build and deploy both succeeded.

B3 Smoke test the live URL

› Open the railway.app URL and check the homepage and every main page load, links work, and there are no console errors on the live deploy. Report anything broken that didn't show up locally.

Prefer deploy-on-push? Connect the GitHub repo in the Railway dashboard instead, then every push auto-builds. The CLI path above is the direct one.

Phase C Connect your domain

Do this after B2. Railway won't show domain options until the service is live and listening, and services don't get a domain automatically. You generate one once Railway sees the

service is listening.

C1 Add the domains and get the records

› Add two custom domains to my Railway service: [your domain] and www.[your domain]. For each one, print the exact records Railway returns, the CNAME target and the TXT verification record, with the host or name and the value spelled out, so I can paste them into my DNS host. Then tell me which record type to use for the bare root versus the www subdomain.

C2 Verify after you set DNS

› I've added the records at my registrar. Check whether [your domain] and www.[your domain] resolve to Railway yet, and whether the SSL certificate has been issued. If it's still pending, tell me exactly what's missing or misconfigured.

READ THIS BEFORE PHASE C: ROOT-DOMAIN GOTCHAS

- ! **You need both records, not just the CNAME.** Railway gives you a CNAME and a TXT record, and both are required. If the TXT is missing, the domain can return a 404 even after the CNAME resolves.
- ! **A bare root domain usually can't take a CNAME.** A root or apex domain (like example.com, or a .md root) often can't hold a CNAME. If your DNS host supports ALIAS, ANAME, or CNAME flattening at the root, use that with Railway's CNAME target. If it doesn't, point www at Railway with a CNAME and redirect the root to www at your host. That's why C1 adds both domains.
- ! **If your host runs a proxy or CDN, turn it off for these records.** Railway routes through its own edge, so a proxy in front causes double-proxying. Disable it for the Railway records.
- ! **Give it time.** DNS propagation plus the SSL certificate can take a while, and some registrars are slow. Don't panic if C2 says pending for a bit.
- ! **Billing.** Railway is usage-based starting around $5 a month, so set up billing or the deploy stays limited.

11 Index and authority

31 Add the basics

> › Add the search verification and analytics to every page.

32 Submit to search

```
$ /seo google
```

> OR SAY Help me submit the site map and request indexing.

Then: do the same in Bing and Yandex.

33 Connect your backlink

> › Update [my repo or main page] to link to [my new site].

34 Check backlinks

```
$ /seo backlinks [url]
```

> OR SAY Check my backlinks and show me the easy wins.

12 Keep it ranking

35 Watch for drops

```
$ /seo drift compare [url]
```

> OR SAY Compare against the baseline and tell me what changed.

36 **Refresh content**

> › Update the oldest pages with fresh info and new dates.

★ Tips

- Run the **/command** when there is one, it's faster and more exact.
- No command? Just say the line. Plain talk works.
- One step at a time.
- If something looks off, just tell Claude what you want instead.
- Swap the example pages, colors, and topics for whatever fits your thing.
- One strong, relevant backlink plus a clean site goes a long way.

› Share freely. Built with Claude Code and claude-seo.