

My Claude Job-Search System

What the five skills do, and how to build them from scratch

This walks you through the job-search system I built inside Claude. It covers the five skills I use, what each one does, and step-by-step instructions to build your own version of each. No coding knowledge needed — if you can write clear instructions in plain English, you can build all of it.

A quick note on the recipes: I've written them so you build *your* system, not a copy of mine. Wherever a skill leans on my personal details — my CV, my figures, my criteria — the instructions tell you to create your own equivalent. The method transfers; the private data stays mine.

First: what a “skill” actually is

A skill is a set of instructions you write once, save, and Claude follows every time you trigger it. Think of it as a recipe card you hand to a very capable assistant. You don't re-explain the job each time — you just say “do the CV one” and it runs the whole recipe.

Under the hood, each skill is a single plain-text file called **SKILL.md**. That file has two parts:

- **A description** — one or two sentences telling Claude *when* to use the skill. This is what makes it fire automatically when you share a job advert, for example.
- **The body** — the actual steps, rules and preferences Claude should follow. This is the bulk of the recipe.

You trigger a skill by typing a slash and its name (for example `/tailor-cv`), or just by describing the task — a well-written description means Claude picks the right skill on its own.

How to create one (the non-technical way)

1. Open a Claude session (I use Cowork).
2. Say: “Help me create a skill that does X.” Describe the job the way you'd brief a colleague — what goes in, what should come out, and any rules that matter to you.
3. Claude drafts the SKILL.md and shows it to you for review. You read it like you'd read a brief: is anything missing, wrong, or too vague?
4. You save it. From then on it's available by name.
5. You improve it over time. Every time the output isn't quite right, you tell Claude what you'd have preferred, and that correction gets folded back into the skill. Mine got good because I kept doing this, not because I got them right first time.

The single most useful habit

Write down your preferences as rules. Not “make the CV good” but “two pages, never three; use the advert's own words; numerals not words for figures.” Specific rules are what separate a skill that saves you time from one you have to redo by hand.

The shared foundation (build this before any skill)

Four of the five skills lean on the same underlying material. Build this first and the skills have something solid to stand on. Skip it and they'll invent things, which is the one behaviour you don't want in a job search.

1. A master achievement bank

This is the backbone. It's one document holding *every* achievement from your whole career — with real numbers, the role it belongs to, and the date. Not a CV. A CV is a selection; this is the full library you select *from*.

Mine holds a clean career timeline, achievements grouped by role, a themed index (so I can pull “everything about events” or “everything about paid media” quickly), and — importantly — a list of the places my own old CVs contradict each other, with a ruling on which figure is the correct one to use. That last part matters more than it sounds: if you’ve been working for years, your numbers drift across versions, and you want one settled source of truth so you’re never caught out in an interview.

Build your own

Gather every CV and appraisal you’ve got. Ask Claude to help you consolidate them into one “achievement bank” document: one entry per achievement, each with a hard number where you have one, grouped by employer and role.

Where two documents disagree on a figure, decide which is right and record the decision. Defensible beats impressive — pick the number you can stand behind if questioned.

Keep it current. When you land a new result in your job, add it straight to the bank so future applications can use it.

2. Your employer-health criteria

Before applying anywhere, I check whether the company is actually healthy. I use three fixed tests, scored out of 3, all readable from LinkedIn:

- Total headcount has grown two years running.
- The marketing team specifically has grown year on year (a growing function means budget and hiring, not cuts).
- Total headcount sits between 500 and 10,000 — my sweet spot for scale without bureaucracy.

Yours will differ. The point is to **decide your tests once**, write them down, and apply them to every company the same way. That consistency is what a skill gives you.

3. Your hard constraints

These are the non-negotiables that no amount of “great role” can override. Mine is a working pattern: hybrid at no more than three days a week in a London office, or fully remote. Anything more is a near-dealbreaker, and I built the scoring so a role that breaks it drops *below* the ones I can actually take, however good it looks. Write yours down — salary floor, location, travel, whatever they are — so the system respects them instead of quietly averaging them away.

The five skills at a glance

Here’s the whole system on one line each. The rest of the document is the full recipe for each.

Skill	What it turns in	What it gives back
<code>/score-job-fit</code>	A job advert	CV-fit score /100, employer health /3, location verdict
<code>/tailor-cv</code>	A job advert + your master CV	A two-page Word CV built for that role
<code>/team-map</code>	A job advert URL	A visual org chart of the hiring team
<code>/job-pitch-video</code>	A job advert + your elevator pitch	A 5-screen slide deck + a timed script to talk over
<code>/weekly-job-scan</code>	Your saved LinkedIn search	A ranked table of the week’s new roles, saved to a living doc

They chain together. `/score-job-fit` is the engine — `/weekly-job-scan` runs it across a whole search, and a good score is what tells me to spend time on `/tailor-cv`, `/team-map` and `/job-pitch-video`. Build them in that order and each one has what it needs.

1. /score-job-fit

The engine of the whole system. Feed it a job and it tells you, honestly, whether the role is worth your time — before you sink hours into an application.

What it takes	A job advert (a LinkedIn job link, or pasted text).
What it gives back	Three separate scores: CV fit out of 100, employer health out of 3, and a location/travel verdict.
Depends on	Your master achievement bank; your three employer criteria; your hard constraints.

Why three scores, not one

Because they answer different questions, and blending them hides problems. A brilliant role at a shrinking company, a dull role at a great company, and a perfect role you physically can't take should each look different. Keeping them apart is the whole point — especially the constraint, which acts as a *gate*, not just another number.

The recipe

1. **Read the job properly.** Open the advert and pull out the responsibilities, the must-haves, and the preferred extras. Read the location, work pattern and travel from the *body of the advert*, never the “hybrid/on-site” tag on the card — that tag is often wrong. If a “hybrid” role doesn't say how many office days, record “days not stated” rather than assuming.
2. **Grab the employer data while you're there.** A LinkedIn job page usually carries a hiring/headcount panel — total employees, company growth, marketing-team growth, size band. One page answers all three employer tests most of the time, so read it before opening anything else.
3. **Map the role against your bank.** Line up the must-haves with real evidence from your achievement bank. Score the must-haves first, the nice-to-haves second.
4. **Score CV fit /100 — honestly.** Name the gaps out loud. An honest 82 you can act on beats a flattering 95. Weigh recency and seniority, not just keyword presence. Keep location out of this number.
5. **Score the employer /3.** Run your three tests against the LinkedIn figures. Cite the actual numbers. Where a figure sits behind a paywall, say so and mark it — never guess a number to complete the score.
6. **Judge location and travel on its own.** OK / soft miss / firm miss against your constraint. A firm miss (too many office days, wrong city, heavy travel) is a near-dealbreaker and gets said plainly.
7. **Report all three together,** with the reasoning and anything you couldn't verify. Default to a short summary in the chat; only save a document if you ask for one.

The rules that make it good

- Honesty over flattery. The score is only useful if it's true. Build “name the gaps plainly” into the skill as a rule.
- Read the advert body for the work pattern, not the card tag.
- Watch for recruiter-posted adverts: the headcount panel then shows the recruiter's data, not the real employer's. Score CV fit only and mark the employer “n/a”.
- Never invent a figure to fill a gap in the score.

2. /tailor-cv

Takes a job advert and your master CV, and produces a tailored two-page CV built to get through the automated CV sift and land with a human. The goal is a CV that reads as if it were written for that one role — because it was.

What it takes	A job advert + your master achievement bank.
What it gives back	A polished two-page Word (.docx) CV, plus a PDF.
Depends on	Your master achievement bank (the single source of every claim).

The idea that makes it work

Tailoring is **reordering and rewording**, not just picking the right achievements. Choosing which bullets to include is the easy half. The tailored CV and the generic one really diverge in how the same material is ordered and phrased. Every claim still comes from your bank — nothing is invented — but it's arranged for *this* advert.

The recipe

1. **Read the advert and note its exact words.** Pull the responsibilities, minimum requirements and preferred qualifications. Note the precise nouns it uses — you'll reuse them.
2. **Map requirements to evidence.** Build a mental table of each requirement against a real achievement. Where you have no direct match, say so honestly rather than papering over it.
3. **Pause and think before writing.** Decide the lead angle, how to frame your current role, and whether any figure in the CV rests on a single source you'd need to defend.
4. **Build the two-page Word document** following your house style (below).
5. **Check it with your own eyes.** Convert to PDF, look at the pages, confirm it's two pages and the layout holds. Never send without looking.

The tailoring moves (the craft)

- **Order bullets by relevance to the advert, not by how impressive they are.** For a field-marketing role, the events bullet leads every role that has one — even if a bigger number sits below it.
- **Order the items inside a bullet the same way.** Lead with what the job cares about most.
- **Use the advert's own nouns.** If it says "partner marketing", write partner marketing, not "partnership marketing". Synonyms cost you the match in automated sifts.
- **Reframe scope honestly to fit the discipline.** The same budget described in the language of the target role reads as a stronger match — without changing the facts.
- **Match the geography to the role.** For an EMEA role, don't dilute with regions the job doesn't care about.
- **Treat the tools list as evidence.** Include the tools that back up the claims elsewhere on the page.

House style — my non-negotiables (set your own)

- Two pages. Never three. If it runs over, cut content — don't shrink the type below 10pt.
- A real Word .docx, not a Google Doc exported to Word — exports balloon to five or six pages and parse badly in applicant-tracking systems.
- Dates right-aligned, so there's clean space between the role title and the date.
- Numerals, not words, for figures — '18 years', not 'eighteen'. Numbers should scan.
- Structure: header → short profile + three bullets → experience → education → tools.
- One settled contact line and one settled set of positioning decisions, so every CV is consistent.

After delivering, the skill names the risks plainly: any single-source figure that made it in, any date or framing that differs from CVs already sent elsewhere, and the obvious interview question the CV invites. No padding, no praise — just what to watch.

3. /team-map

Takes a job URL and produces a visual map of the team behind the role: the likely hiring manager, the reporting line, and every relevant person's career history — plus the patterns that fall out of it. It turns "I'm applying into a black box" into "I know who's who, who I'd report to, and what this team tends to hire."

What it takes	A job advert URL (LinkedIn or a company careers page).
What it gives back	A visual org chart (an interactive web page) of the hiring team, with career histories and patterns.
Depends on	A logged-in LinkedIn session in your own browser, driven through Claude.

One rule to build in from the start

This is passive reading of public LinkedIn profiles only, through your own logged-in browser. The skill never sends connection requests, never messages anyone, and never clicks anything that acts on your account. Bake that limit into the skill so it can't drift.

The recipe

1. **Parse the job.** Pull the exact role title, the team/function, the segment (SMB, Mid-Market, Enterprise...), the region, and any named manager. Most adverts don't name the manager — that's expected; you'll infer it. Note segment and region precisely; they're what tell you which sub-team owns the role.
2. **Find the team on LinkedIn.** Run a people search built from company + function + region. Read the results, and note everyone whose current role is at the target company in the relevant function. Run a second and third search with varied wording to catch people the first missed.
3. **Pull each person's career history.** For each person, open their full experience list and capture their current title, start date and tenure, and the chain of previous employers.
4. **Analyse it against the role.** Work out the likely hiring manager (the lead of the sub-team that owns this segment), sketch the reporting line, and flag anyone whose role is global rather than regional. Mark inferred lines as inferred — never assert an unconfirmed reporting line as fact.
5. **Find the patterns.** Shared former employers (a feeder pipeline into this team), clusters of recent joiners (a fast build-out), the mix of backgrounds they hire. These are the real prize — they tell you what the team values and who to reach out to.
6. **Build the map.** Produce a single clean web page: a headline naming the role, a few "pattern" tiles with the key numbers, then the org chart — department head at the top, segment leads as branches, team members as cards underneath. Each card shows the person's name, title, location, tenure, and their previous employers.

It closes by stating which reporting lines are confirmed and which are inferred, and citing the date the profiles were read. LinkedIn is self-reported and sometimes stale, so the skill presents findings evenhandedly and says so where the data is thin.

4. /job-pitch-video

Takes a job advert and your elevator pitch, and turns them into a short slide deck plus a timed script. You screen-record yourself talking over it — a 60–90 second video you send to the hiring manager to stand out. The skill builds the deck and writes the words; you provide the face and voice.

What it takes	A job advert + an elevator pitch you write first.
What it gives back	A five-screen slide deck (a web page) + a timed script marked slide by slide.
Depends on	Your master achievement bank (for the proof points).

The one editorial rule that matters

Three proofs, not three topics. The instinct is “why me / why the company / why the role”. Those are cover-letter sections and they excite nobody in 90 seconds. Convert them into evidence:

- “Why me” becomes: I’ve already run this exact job, recently, with numbers.
- “Why the company” gets cut entirely — it’s flattery every applicant writes, it eats a fifth of your runtime, and it sets you apart from no one.
- “Why the role” becomes: the way I work, proved by something I actually did.

And don’t raise your gaps. If there’s an obvious mismatch, the video is no place to plant a doubt they haven’t raised. Prepare that answer for interview instead.

The recipe

1. **Get both inputs.** The advert (note its exact words) and the elevator pitch. Treat the pitch as raw material, not a finished script.
2. **Budget the words.** 60–90 seconds is 150–225 spoken words. A written pitch is usually double that. Half has to go.
3. **Build five screens.** Cover → Proof 01 (the most job-relevant recent evidence, as big numbers) → Proof 02 (sector and segment fit) → Proof 03 (how you work, with the one memorable line) → Close (a three-word summary, one warm line, your contact details).
4. **Rank the proof by relevance, not size.** The most recent thing that is literally the job on the tin leads, even when a bigger number sits behind it.
5. **Brand it to the company — carefully.** Use their colours and layout conventions so it reads as someone who knows their world. Never use their logo (it’s a trademark, and a forwarded deck could look company-authored). Pick the closest free font to theirs and say so.
6. **Deliver the deck and a timed script** marked slide by slide, with the timing stated (“about 70 seconds”), so you can read it off a second screen while you record.

The style notes (learned from my own edits)

- Slide titles are plain noun phrases, not clever sentences. Every title takes a full stop.
- Numbers are the heroes — big, in the accent colour. Everything else stays quiet.
- No captions or footnotes under slides. If it matters, it goes in the script, not on the page.
- It opens cold — no “hello” — because it comes after your intro and the org chart.

5. /weekly-job-scan

The time-saver. Once a week it opens my saved LinkedIn search, takes every role posted in the last seven days, scores each one, and writes a ranked table into a living document. Instead of scrolling job boards for an hour, I get a shortlist that's already scored and sorted.

What it takes	Your saved LinkedIn jobs search (your preferences, your filters).
What it gives back	A ranked table of the week's new roles, added to the top of a living document that keeps its history.
Depends on	The /score-job-fit skill (it reuses those rules) and a logged-in LinkedIn session.

The recipe

1. **Open your saved search** in your logged-in browser and set the date filter to "Past week". Confirm the filter actually took before going on.
2. **Collect the jobs the way a person does.** Page through the numbered pages of results — don't try to grab the whole list in one scroll (LinkedIn loads it in pieces and bulk-scraping stalls). Capture each role's title, company, location, posting age and any salary, and de-duplicate.
3. **Score each job one at a time**, on its own page, using the /score-job-fit rules — CV fit, employer health, and location. Read your achievement bank once at the start and reuse it for all of them.
4. **Fold in a level check.** Dock roles that sit clearly below your level so a junior seat can't out-rank a senior one, and flag any hard gate (a required language or industry you don't have) rather than ranking it as if you qualify.
5. **Compute one combined score.** Blend CV fit and employer health (I weight them 70/30), then apply location as a gate — a role that breaks my working-pattern constraint gets points subtracted so it drops below the roles I can actually take. Keep the individual scores visible in their own columns so nothing hides behind the blend.
6. **Write it to a living document.** One table per week, sorted by combined score, added above last week's so the history builds up. Columns: rank, role, company, location, CV fit, level, employer, combined, salary, and a direct link. Under it, a couple of plain lines: how many were scanned, anything that blocked a score, and the two or three worth acting on first.

Make it run itself

I have this set to run automatically every Monday as a scheduled task, so the shortlist is waiting for me at the start of the week. You can set the same thing up — a recurring task that triggers the scan on a schedule — so the system works even when you don't.

One refinement worth adding

Right now the scan adds new roles but leaves old ones in the table. A small addition to the skill — "remove listings that are no longer live" — would keep the list clean. This is a good example of how these skills improve: you spot a rough edge in use, and you fold the fix back into the instructions.

If you're building this from scratch

Here's the order I'd build it in, so each piece has what it needs when you get to it.

1. **Build the master achievement bank first.** Everything else draws on it. This is the biggest single job, and the most valuable — it's worth doing properly even before you touch a skill.
2. **Decide your employer criteria and your hard constraints** and write them down. A short paragraph each is enough.
3. **Build /score-job-fit.** It's the engine, and building it forces you to make your criteria concrete.
4. **Build /tailor-cv.** Now that the bank and the scoring exist, this has everything it needs.
5. **Build /team-map and /job-pitch-video.** These are the "stand out" tools you use once a role scores well enough to be worth the effort.
6. **Build /weekly-job-scan last,** because it reuses the scoring skill. Then set it on a schedule.

The two principles behind all of it

1. One source of truth. Every claim comes from the achievement bank — never invented, never freshly written. That's what keeps you honest and consistent across dozens of applications.
2. Improve by correction. None of these worked perfectly first time. They got good because every time the output missed, I told Claude what I'd have preferred, and that became a new rule. Build a rough version, use it for real, and fix it in use. That's the whole method.

Happy to walk you through any of these live if it helps.