Claude LinkedIn Agent Skill Setup Guide

Install 11 free Claude skills that draft your LinkedIn posts, comments, and replies, then run a local humanizer that scores your draft against five AI detection checks.

Why this is different from a prompt

The LinkedIn Agent Skill is a repo of 11 separate Claude Skills built by Jake Schincariol, released under the MIT license. Each skill owns one job: writing posts, drafting comments, sorting replies, scoring your profile, planning your week, humanizing a draft, writing carousel copy, repurposing a piece of content into a week of posts, drafting DMs, triaging your inbox, and auditing past posts.

A single giant prompt tries to do all of that at once and drifts. Splitting it into 11 skills means Claude reads only the one file that matters for the job you asked for, and every one of them reads the same voice file first so the output sounds like you instead of like each other.

None of the 11 skills post to LinkedIn for you. Every one stops at a draft and waits for you to review and paste it in yourself. That is by design: automating the LinkedIn site with a browser or a third party tool breaks LinkedIn's User Agreement, so the repo never touches the site directly and needs no LinkedIn API key. The README puts it in one line: nothing gets posted until you say yes.

01

How it actually works

1. You call a skill by its slash command, for example /li-post to draft a post or /li-comment to draft a comment on someone else's post. /li-comment covers nine different comment types, so the draft changes depending on whether you are agreeing, adding a data point, or asking a question.
2. The skill reads templates/voice.md first, a file you fill in once with your tone, background, and a few sample lines, so every draft sounds like the same person, whether it came from /li-post, /li-reply, or /li-dm.
3. The skill hands you back a draft. For /li-post that draft is built from one of 21 hook formulas; for /li-human it is a cleaned version of text you already wrote, plus a score; for /li-audit it is a breakdown of your past posts by engagement metrics instead of a new draft at all.

---

02 The one thing almost everyone gets wrong

WATCH OUT

Skipping templates/voice.md and running /li-post straight away gives you a technically fine post that reads like nobody in particular. The README calls this out directly: spend ten minutes on the voice file first, either filling it in by hand or handing Claude three of your own past posts to extract your voice from. Every one of the 11 skills reads that same file before it drafts anything, so this ten minutes pays off across all of them, not just one.

---

03 The files

SOURCE CODE

The full repo:

https://github.com/nessalazne/linkedin-agent-skill

The pieces worth knowing before you install, straight from the repo structure:

SKILLS/LI-POST/HOOKS.JSON

```
21 hook formulas with templates and examples, used by /li-post to draft the opening
line of a post.
```

SKILLS/LI-HUMAN/SLOP.JSON

```
113 stock AI terms, 17 invisible character classes, and 11 structural tells that /li-
human checks a draft against.
```

SKILLS/LI-HUMAN/HUMANIZE.PY

```
python3 humanize.py draft.txt --report
# runs the three-pass clean: invisible characters, typography, then the 113-term
lexicon swap
```

SKILLS/LI-HUMAN/DETECT.PY

```
python3 detect.py draft.txt
python3 detect.py before.txt after.txt
# scores 0-100 across burstiness, specificity, slop density, fingerprint, and voice
```

SKILLS/LI-PROFILE/RUBRIC.JSON

```
The 12-part, 100-point rubric that /li-profile scores your profile against.
```

TEMPLATES/VOICE.MD

```
The voice file every one of the 11 skills reads before drafting. Copy it to
~/.claude/linkedin/voice.md and fill in your own tone, background, and sample lines.
```

Set it up in Claude Code or Codex

1. In Claude Code, the skill folder goes at ~/.claude/skills/. In Codex, it is your project's own folder, or .codex/skills/linkedin-agent-skill/ with a pointer added to AGENTS.md.
2. The folder name has to match the name: field inside each skill's SKILL.md, so if you rename a folder after cloning, open SKILL.md and rename the name: field to match.
3. Clone the repo: git clone https://github.com/nessalazne/linkedin-agent-skill.git
4. Copy the skills in: cp -r linkedin-agent-skill/skills/li-* ~/.claude/skills/ (swap that destination for a project's .claude/skills/ if you want it local to one repo instead of global). If you would rather skip the manual copy, paste the repo URL into Claude and ask it to install the skill, or run /plugin marketplace add nessalazne/linkedin-agent-skill followed by /plugin install linkedin-agent.
5. Copy templates/voice.md to ~/.claude/linkedin/voice.md and fill it in, either by hand or by pasting three of your own LinkedIn posts and asking Claude to extract your voice from them.
6. No API key setup is needed anywhere in this repo. Every skill runs locally and /li-human explicitly uploads nothing.
7. Test the install with a low-stakes command first: ask Claude to run /li-post on a topic you already know well, and check that the draft reads like your voice file before you trust it with anything real.
8. A successful first run looks like a draft post that mentions specifics from your voice file, followed by a hook formula name if you ask which one it used. If it comes back generic, go fix templates/voice.md before you run anything else.

💡 QUICK TIPS

- Run /li-human on a few of your own past posts before you write anything new, just to see what your baseline detector score already looks like.
- Keep before.txt and after.txt versions of one draft around so python3 detect.py before.txt after.txt has something real to compare.

04

The commands I actually run

DRAFT A POST

```
/li-post [TOPIC], use a hook formula that fits a [AUDIENCE] audience
```

CLEAN AND SCORE A DRAFT

```
python3 skills/li-human/humanize.py [DRAFT_FILE].txt --report
python3 skills/li-human/detect.py [DRAFT_FILE].txt
```

SCORE MY PROFILE

```
/li-profile, review my LinkedIn profile against the rubric and tell me the 3 lowest-scoring sections
```

TURN ONE ASSET INTO A WEEK OF POSTS

```
/li-repurpose [SOURCE_CONTENT], turn this into 5 LinkedIn posts for next week
```

PLAN THE WEEK

```
/li-plan, build my posting and engagement plan for the week of [DATE]
```

DRAFT A COMMENT

```
/li-comment on this post: [PASTE_POST_TEXT]. Write a comment that adds a real data point, not just agreement
```

AUDIT LAST MONTH

```
/li-audit, look at my last [NUMBER] posts and tell me which 3 got the most engagement and why
```

When it breaks

ERROR	FIX
/li-post gives a generic draft that does not sound like you	templates/voice.md is empty or too thin. Fill it in by hand or hand Claude three real posts of yours and ask it to extract your voice first.
A slash command is not recognized	The skill folder name does not match the name: field in that skill's SKILL.md. Open SKILL.md and rename the field, or rename the folder to match.
detect.py reports a low score even after humanize.py ran	humanize.py only fixes the three automatic passes. Structural tells like rule-of-three lists, one-word rhetorical questions, and hashtag walls are flagged, not auto-fixed. Rewrite those lines by hand.
A skill installed globally is not showing up inside one project	Global skills live in ~/.claude/skills/. For a project-local install, copy the same skill folders into that project's own .claude/skills/ instead.
Codex cannot find the skill	Codex reads .codex/skills/ or a pointer in AGENTS.md, not ~/.claude/skills/. Copy the folder there or add the pointer.
/li-comment or /li-reply drafts feel off-topic	Those skills draft from the post or comment you paste in. Paste the actual text you are replying to, not just a summary of it.

Ready to go deeper?

This playbook is the short version. Inside the hub you get the full build walkthroughs, the prompt library, and the systems I use to ship this in a weekend.

Join the Build →

https://hub.digicuratoragency.com/join