The LinkedIn OS Setup Guide

Install 11 free Claude skills that draft your LinkedIn posts, comments and replies, and wait for your approval before anything goes out.

What this actually is

Serge Bulaev published a free bundle of 11 Claude Skills that covers a full LinkedIn workflow. It is MIT licensed, there is no signup, and it runs inside Claude Code, claude.ai, Claude Desktop, Codex, or any agent that reads **SKILL.md** files.

The reason I bothered with this one is the structure. Most LinkedIn AI tools are a single prompt wearing a product name. You paste your topic in, you get a post out, and it sounds like every other post that tool made that day. There is no memory of your voice and no separate handling for a comment versus a reply versus a profile rewrite.

This bundle splits the work into 11 separate skills, each with its own instruction file. When you ask Claude for a comment, it reads the comment skill, which knows a comment should land at 200 to 350 characters and carry one sharp point. When you ask for a post, a different file takes over with a 900 to 1,300 character target and 20 hook formulas to pick from. Same agent, different playbook per job.

Setup is one command. Everything below is what I would tell a friend who wanted it running tonight.

01

How it actually works

1. A skill is a folder with a **SKILL.md** file inside it. That file is plain English instructions telling the agent how to do one job, plus the rules it must follow. There is no code to run and nothing to configure.
2. When you ask for something LinkedIn shaped, Claude matches your request against the description at the top of each skill file and loads the one that fits. You do not name the skill, you just describe what you want.
3. The skill drafts, shows you the result, and stops. Publishing only happens if you connect Publora and say yes to that specific draft. Out of the box the bundle writes things and hands them to you.

02

The one thing almost everyone gets wrong

People install all 11 skills, use the Post Writer once, decide the output sounds like AI, and quit. The Post Writer is the first half of the system. The Humanizer is the second half, and skipping it is why the output sounds generated.

WATCH OUT

Run every draft through the Humanizer before you read it as finished. It strips em dashes, the AI vocabulary set (leverage, delve, harness, streamline, foster, unlock, fundamentally), and rule-of-three lists. Then it runs the text through GPTZero, Originality.ai, ZeroGPT, Sapling and Copyleaks and reports the spread, so you get real scores instead of a promise. Writing then checking is the whole loop. One without the other is why people give up on this.

03

The files

SOURCE CODE

The full repo, MIT licensed, no signup:

https://github.com/sergebulaev/linkedin-skills

You do not have to touch any of this to use the bundle. I am showing it so you know what landed on your machine and where to look when you want to change a rule.

WHAT IS IN THE REPO

```
linkedin-skills/
  skills/
    linkedin-post-writer/        SKILL.md
    linkedin-comment-drafter/    SKILL.md
    linkedin-reply-handler/      SKILL.md
    linkedin-humanizer/          SKILL.md
    linkedin-hook-extractor/     SKILL.md
    linkedin-content-planner/    SKILL.md
    linkedin-thread-monitor/     SKILL.md
    linkedin-engager-analytics/  SKILL.md
    linkedin-profile-optimizer/  SKILL.md
    linkedin-employee-advocacy/  SKILL.md
    linkedin-repurposer/         SKILL.md
  lib/
    url_parser.py                LinkedIn's 3 URL formats
    apify_client.py              reads posts, comments, engagers
    publora_client.py            publishing actions
    pixfaro_client.py            illustrations
  references/
    founder-topics.md            10 founder angles
    voice-profile.md             your voice and brand fields
    industry-benchmarks.md       engagement rates by industry
  .env.example
  requirements.txt
```

If you want any of the optional API keys, this file goes in the **linkedin-skills/** root. The repo ships a **.env.example** you can copy instead of typing it out.

.ENV

```
# Only add the ones you actually want. All four are optional.

# Publish straight to LinkedIn (free tier: 15 posts/month)
PUBLORA_API_KEY=sk_paste_your_key_here
LINKEDIN_PLATFORM_ID=linkedin-paste_your_id_here

# Read real post bodies, comment threads and engagers (free tier: $5/month credit)
APIFY_TOKEN=apify_api_paste_your_token_here

# Generate an illustration for a post
PIXFARO_TOKEN=pf_live_paste_your_token_here
```

Running an agent that does not auto-discover skills, like OpenClaw? Clone the repo into your working directory and paste this into your system prompt.

SYSTEM PROMPT ADDITION (OPENCLAW AND SIMILAR)

```
You have LinkedIn marketing skills in ./linkedin-skills/.
For any LinkedIn task, read the relevant skills/*/SKILL.md first.
Use lib/url_parser.py for URL parsing,
    lib/apify_client.py for reading posts / comments / engagers,
    lib/publora_client.py for publishing actions.
```

OPTIONAL SERVICES, ONLY IF YOU WANT THEM

- Apify, reads real post bodies and comment threads
  https://console.apify.com/sign-up
- Publora, publishes straight to LinkedIn
  https://app.publora.com/signup
- Pixfaro, generates post illustrations
  https://pixfaro.com

Set it up in Claude Code or Codex

1. **Pick your install line.** In Claude Code (CLI, VS Code or JetBrains) run **/plugin marketplace add sergebulaev/linkedin-skills** and then **/plugin install linkedin-skills@linkedin-skills**. On claude.ai, open Skills in the sidebar, click Add from GitHub, and paste **sergebulaev/linkedin-skills**. In Codex CLI run **codex plugin marketplace add sergebulaev/linkedin-skills** then **codex plugin add linkedin-skills@linkedin-skills**. On any other agent, run **npx skills add sergebulaev/linkedin-skills**.
2. **Or clone it, if you want to edit the rules.** Run **git clone https://github.com/sergebulaev/linkedin-skills.git**, then **cd linkedin-skills** and open that folder as your working directory. Claude Code picks up the skills folder automatically. This is the version to use if you plan to rewrite the voice rules to match how you actually talk.
3. **Start a new conversation.** Skills load when a session starts, so an already-open chat will not see them. This is the single most common reason someone thinks the install failed.
4. **Test it with no keys at all.** Ask for a LinkedIn post on a topic you know well. If a draft comes back, you are done. Everything past this point is optional.
5. **Add API keys only when you hit the limit.** Create a file called **.env** in the linkedin-skills root and put the keys in it, or copy **.env.example** and fill in the blanks. If you are not comfortable editing files, ask Claude to create the .env file and paste the key in for you. Then install the two packages the clients need with **pip install requests python-dotenv**.
6. **Verify a Publora connection before you trust it.** Ask for a test post scheduled 24 hours out, confirm you get a scheduled-post ID back, then cancel it in the Publora dashboard. Do this once so you are not discovering an auth problem on a post you cared about.

> 💡 **QUICK TIPS**
>
> - The folder name has to match the **name:** field inside each SKILL.md. If you rename a folder by hand, that skill stops loading.
> - Start with no API keys. Four of the skills fall back to asking you to paste the post text, which is fine while you work out whether you like the drafts.
> - Add Apify first if you comment a lot. Reading the real thread is where the drafts get noticeably better, and the free tier covers a normal week.
> - If **pip install** fails, use a virtual environment: **python -m venv venv**, then **source venv/bin/activate**, then install.

04

The commands I actually run

DRAFT A POST

```
Write me a LinkedIn post about [YOUR TOPIC]. I want it to [DRIVE COMMENTS / BUILD
AUTHORITY / GET PROFILE VIEWS]. My audience is [WHO THEY ARE].
```

SCRUB THE AI TELLS, THEN CHECK

```
Humanize this post, then run it through the detector spread and show me the scores:

[PASTE YOUR DRAFT]
```

STEAL THE STRUCTURE OF A POST THAT WORKED

```
What hook formula does this post use? [LINKEDIN POST URL]

Give me the blank template, then fill it with my topic: [YOUR TOPIC].
```

COMMENT WITHOUT SOUNDING LIKE A BOT

```
Draft a comment on this post: [LINKEDIN POST URL]

My angle: [THE ONE POINT YOU WANT TO MAKE]. Keep it under 350 characters.
```

PLAN THE WEEK

```
Create a 7-day LinkedIn content plan. I am a [YOUR ROLE] targeting [YOUR AUDIENCE].
Include daily topics, formats, hooks, posting times, and which posts to comment on.
```

When it breaks

ERROR	FIX
Skills never activate when I mention LinkedIn	Start a new conversation. Skills load at session start, so an open chat will not see a fresh install.
Installed but nothing shows in the Skills panel	Check the install actually ran through the panel, /plugin install, or codex plugin add. A plain download into a random folder does not register.
Publora API key not provided	The .env file is missing or in the wrong folder. It belongs in the linkedin-skills/ root, not your home directory.
401 Unauthorized from Publora	The key expired. Go to Publora Settings, then API, then create a new key and replace it in .env.
404 on a comment or post	Your LINKEDIN_PLATFORM_ID is wrong. Copy the full string from Publora Channels, including the linkedin- prefix.
400 reactionType error	A known Publora quirk. The skills handle it. If you are calling the API by hand, use PRAISE instead of CELEBRATE and INTEREST instead of INSIGHTFUL.
pip install fails	Use a virtual environment: python -m venv venv, then source venv/bin/activate, then pip install requests python-dotenv.
The drafts still sound like AI	You skipped the Humanizer. Run every draft through it and read the detector scores before you decide the bundle does not work.

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