il

INBOUNDLI

The LinkedIn Lead System

Built so you can post consistently, grow the audience you want to reach, and bring in more leads for your business.

Khyati Hooda, InboundLi

inboundli.com

START HERE

What this is

This is the system I built to keep my own LinkedIn working without it taking over my week. It does the research, the drafting, and the outreach. The review and the posting stay with me.

Some of this will sound familiar:

- → You post rarely, because real work always wins the week.
- → You post consistently but it doesn't get engagement.
- → You post and get engagement but no clients come from it.
- → Your outreach is manual, so it stops the moment you get busy.
- → Your pipeline runs on referrals and whoever happens to find you.
- → You've watched people worse than you win clients off LinkedIn and can't work out what they're doing.
- → You know LinkedIn should be a lead source, and right now it's just a place you have a profile.

If any of that lands, this document is worth your time.

How to use it

The system has three parts: a weekly research workflow that pulls source material, a chain of Claude skills that turn that material into finished posts, and a daily engagement workflow that puts you in front of the right people. Each part works on its own. You can build the research workflow first and get value from it before the rest exists.

To build it, hand this document to Claude. It contains the actual instruction files for every skill. Tell Claude who you are and who you want to reach, then have it build each skill and set up the two workflows.

What this document covers, in order

- → The LinkedIn rules the system is built around, and the design decision each one drove.
- → The weekly research workflow that pulls your source material.
- → The Claude skill chain that turns that material into finished posts.
- → The engagement workflow and the comment skill.
- → The tools you need and what they cost.

> **One honest note before you start.** This is the complete system, not an overview. Building it takes real time. You will need Apify, Sales Navigator, n8n, and Claude, and you will spend hours setting up the watchlist, finding the right people, and tuning the skills to your voice. If you have that time, everything you need is here. If you would rather have it built and run for you, the last page tells you how to reach me.

SECTION 01

The rules the system is built around

These come from LinkedIn's published research on its 360Brew ranking model and from independent reach analyses. Each rule below changes how far a post travels. What matters here is not the rule on its own, but the design decision it drove in the system. The system is the way it is because of these rules.

Consistency is a ranking signal

The ranking model treats a steady posting rhythm as a sign of an active account. An irregular account gets shown to fewer people even when individual posts are good.

Design decision: the system produces a full week of posts in one run, so posting daily never depends on having ideas or time on a given day. The work that needs consistency is removed from the days that need it.

The first line carries the post

The first line is the only part of a post that both the reader and the ranking model see before deciding whether to expand it. A setup sentence instead of the actual point loses the reader and reads as low signal to the model.

Design decision: the hook gets its own stage in the chain. Every post's first line is audited on its own, against its own rubric, and rewritten if weak.

Most readers are on a phone

A block of text longer than three lines on mobile gets scrolled past.

Design decision: the drafter writes in LinkedIn format from the first line. Short paragraphs, line breaks, arrow bullets. Posts are never written as articles and reformatted later.

Reading level changes reach

Posts written above a tenth grade reading level get meaningfully less reach. The words your buyer uses for their own problem travel further than the words you use to sell the solution.

Design decision: a hard reading-level cap is built into the drafter's voice rules and checked again at the final review. Buyer language, not seller language.

Specific content gets matched to specific readers

The ranking model can match a specific post to a specific audience. A specific topic gives it something to work with. A vague one gives it nothing, so it travels nowhere.

Design decision: the content planner builds each post around one single message, not a broad theme. One post says one thing, specifically.

Raw AI output is penalised

Posts that read as entirely AI-written get less reach and far less engagement. The system detects it.

Design decision: the chain uses AI to draft and edit, but a dedicated review stage strips AI cadence, fragment rhythm, and scenic openings. The substance and the specific examples are yours.

The first 60 minutes decide how far a post goes

The model decides how far to push a post based on the engagement it gets in roughly the first 60 to 90 minutes. You generate that signal by being in your buyers' comment sections right before and after you publish, so your name is already in their feed when your post lands.

Design decision: this is why the system has a separate daily engagement workflow and a comment skill. Posting is only half of it. The comment skill drafts what you say during that window.

Comments are ranked on what they add

A one-word comment does nothing. A comment that adds a real point, a specific number, or a sharp question is read as high-signal and earns reach for you as well as the original poster.

Design decision: the comment skill never drafts a compliment. Every comment adds a point, shares an observation, or asks one real question.

Links in the post body pull reach down

A link preview card sitting in the post body lowers reach. The value should be in the post; the link, if any, goes in the first comment.

Design decision: posts in this system deliver their value in full. Lead-magnet posts drive a DM keyword, not an outbound link.

> The rules about timing, commenting, and consistency are yours to hold. The system writes the post and drafts the comment. It does not press publish. That stays with you, on purpose.

SECTION 02 The weekly research workflow

The system writes posts from real source material, not from a blank page. That material comes from a workflow that runs once a week. It reads a watchlist of people worth learning from, pulls their recent posts, and saves them to a file the skill chain reads.

What it collects

It pulls the latest posts from each account on your watchlist. It does not filter at this stage. The job here is to gather the raw posts in one place. The filtering happens later, inside Claude.

The flow

```
Manual trigger
  -> Read watchlist sheet
  -> Clean URLs and pull a username from each
  -> Scrape posts via Apify (one account at a time)
  -> Combine into one file
  -> Save dated JSON to Google Drive
```

Step by step

1. Read the watchlist. A Google Sheet with one column of LinkedIn profile URLs. You add and remove names as you find better accounts. The sheet is the only part you touch by hand.

2. Clean the URLs. A code node strips tracking parameters, drops anything that is not a profile URL, removes duplicates, and pulls a clean username out of each URL. The Apify actor wants a username, not a full URL, so this step matters.

3. Scrape the posts. An HTTP node calls the Apify actor `apimaestro/linkedin-profile-posts` once for each account and asks for the latest 10 posts.

4. Combine the results. A second code node flattens every response into one list, keeping author, timestamp, post text, and engagement count. It throws an error if zero posts came back, so a silent failure never reaches Claude.

5. Save to Drive. The combined list is written to a dated JSON file and uploaded to a Google Drive folder. Claude reads from that folder when you run the content pipeline.

> **The watchlist starts empty. You build it.** The workflow cannot guess who you want to learn from. To build your first watchlist, find people who post about LinkedIn growth, content, and winning clients, who post regularly, and whose posts get strong engagement for their follower count. Start from one person you respect in this space, look at who comments on their posts and whose posts they comment on, and follow that web outward. Two hops gives you 30 to 40 accounts. Keep an account only if its engagement is strong relative to its follower size. A small account pulling heavy engagement is better signal than a large one pulling little.

The clean-username fix. If you build this workflow from scratch, the one place it commonly breaks is the URL handling. The Apify actor's username field wants a clean username slug, not a full URL. The code node must extract it: match `linkedin.com/in/<slug>` and pass only the slug. The corrected workflow files ship alongside this document.

SECTION 03 The Claude skill chain

A skill in Claude is a folder with an instruction file and any supporting documents. When you give Claude a matching command, it reads the skill and follows its rules. These skills run as a chain, orchestrated by one parent skill. Each stage writes a file the next stage reads, so no step gets skipped and a broken input never reaches a finished post.

The chain produces LinkedIn posts only. It runs six stages, then stops for your review. Nothing posts automatically.

The chain at a glance

```
linkedin-content-pipeline   (orchestrator)
   |
   Stage 1   post-curator        watchlist-posts.json -> 01-curated-posts.json
   Stage 2   content-planner     01-curated-posts.json -> 02-content-plan.json
   Stage 3   framework-picker    02-content-plan.json -> 03-frameworks.json
   Stage 4   linkedin-drafter    + 03-frameworks.json -> 04-drafts.json
   Stage 5   hook-editor         04-drafts.json -> 05-hook-fixed.json
   Stage 6   post-reviewer       05-hook-fixed.json -> final-posts.md
   |
   HARD STOP -> you review final-posts.md and post manually
```

The content mix

Every weekly run produces 7 posts to a fixed mix. The planner builds to it; the reviewer checks it held.

- → **3 posts** on lead generation: how a system or workflow brings clients in. The hero theme.
- → **2 posts** in buyer language: a real problem named in the reader's own words, then reframed.
- → **1 post** with a contrarian take on LinkedIn growth or winning clients.
- → **1 lead-magnet post** that ends with a comment keyword driving DMs.

How to build the skills

The section that follows contains the actual instruction file for every skill in the chain. To build the system, create a folder for each skill and save its instruction file as `SKILL.md` inside. Then point Claude at the folders. The orchestrator calls the rest in order.

> **Set the writer and the reader first.** Before Claude builds a single skill, tell it who the writer is and who the reader is. The writer is you. The reader is the person you want as a client. Every voice rule and every audit in this system depends on those two facts being set.

SECTION 04

The skill files

What follows is the actual instruction file for every skill in the chain. To build the system, create a folder for each skill and save its file as `SKILL.md` inside, then point Claude at the folders. The skills appear in the order they run.

linkedin-content-pipeline ORCHESTRATOR

```
---
name: linkedin-content-pipeline
description: Khyati Hooda's content pipeline orchestrator for InboundLi. Runs a 6-stage chain that
turns a weekly watchlist scrape (watchlist-posts.json) into 7 finished LinkedIn posts ready to ship.
Stages run in sequence, each writing a state file the next stage reads. Hard stops after the final
stage for human review. Nothing posts automatically. Use when Khyati says "run the pipeline",
"generate this week's posts", or "build content". Self-contained orchestrator that calls post-
curator, content-planner, framework-picker, linkedin-drafter, hook-editor, post-reviewer.
---

# linkedin-content-pipeline

Orchestrator for the weekly content chain. This skill is a router. It calls each stage skill in
order and confirms one stage's output file exists before invoking the next.

The pipeline produces LinkedIn posts only. There is no X/Twitter drafting stage.

## When to use

- "run the pipeline"
- "generate this week's posts"
- "build content"
- "build LinkedIn posts for the week"

## What it produces

A weekly bundle of 7 finished LinkedIn posts in `final-posts.md`, ready for the reader to review and
post manually across the week.

## Hard rules

1. **Never skip a stage.** Every stage produces a state file. The next stage reads that file.
2. **Never run stages in parallel.** Sequential only.
3. **One human stop.** After the final stage, STOP and present the result. Never post anything.
4. **Token budget per stage:** roughly 3,000-8,000 output tokens per stage. If a stage produces far
more, the prompt is wrong.
5. **No silent fallback.** If an input file is missing or stale, STOP and say so. Never fall back to
internal knowledge or invent source material.

## Input required before a run

`watchlist-posts.json` — the weekly scrape produced by the n8n weekly content workflow. It must be
recent (within the last 14 days).

If it is missing or stale → STOP. Tell the reader: "I need a fresh watchlist-posts.json. Run the
weekly n8n content workflow first, then upload the file it saves to Drive."

## Pipeline stages (run in this exact order)
```

```
Stage 1 — post-curator
   reads:  watchlist-posts.json
   writes: 01-curated-posts.json
   job:    drop posts that cannot help the reader (personal updates,
           motivational quotes, engagement bait, job posts, off-topic
           posts). Keep the rest with a note on what is reusable.

Stage 2 — content-planner
   reads:  01-curated-posts.json
   writes: 02-content-plan.json
   job:    extract 7 post ideas for the week. Each idea carries one
           single message, the signal from the curated posts that
           points to it, and a research note. Enforces the content
           mix (see below). Anti-paraphrase rule: an idea is a topic
           territory, never a reworded version of one source post.

Stage 3 — framework-picker
   reads:  02-content-plan.json
   writes: 03-frameworks.json
   job:    assign ONE writing framework to each of the 7 ideas, from
           the framework library inside the framework-picker skill.
           Picks on fit, not variety.

Stage 4 — linkedin-drafter
   reads:  02-content-plan.json + 03-frameworks.json
   writes: 04-drafts.json
   job:    write the 7 LinkedIn posts. LinkedIn format from the start:
           short paragraphs, line breaks, a hook under 80 characters.
           Writes from the idea and the research, never from a source
           post directly. One message per post.

Stage 5 — hook-editor
   reads:  04-drafts.json
   writes: 05-hook-fixed.json
   job:    audit the first line of every draft. Rewrite weak hooks.
           Confirm each hook's promise matches what the post delivers.

Stage 6 — post-reviewer
   reads:  05-hook-fixed.json + 01-curated-posts.json
   writes: 06-final.json + final-posts.md
   job:    final audit on an 8-point evidence-required rubric. Auto-
           rewrites posts failing 3-5 checks. Kills posts failing 6+
           or any paraphrase check. Writes the ship-ready bundle.
           HARD STOP HERE.
```

Content mix (enforced at Stage 2)

The 7-post week is planned to this mix. It is not a suggestion; the content-planner builds to it.

- **3 posts** — LinkedIn lead-generation: how a system, workflow, or piece of automation brings clients in. The hero theme.
- **2 posts** — buyer-language posts: name a real problem the reader has, in their own words, then reframe it.
- **1 post** — a contrarian take on LinkedIn growth or how people get clients.
- **1 post** — a lead-magnet post: ends with a comment CTA that drives DMs (only if the giveaway asset it points to actually exists).

If 7 does not divide evenly in a given week, the planner favours the lead-generation theme.

```
## Execution protocol

For each stage:
1. Verify the input file exists. If not → STOP, report which file is missing.
2. Invoke the stage skill.
3. Verify the output file exists after the stage runs. If not → STOP, report which stage failed.
4. Move to the next stage.

## What to show at the hard stop (after Stage 6)

1. The path to `final-posts.md`.
2. A one-line summary: "Pipeline complete. Stage 1 curated X posts. Stage 6 passed Y of 7 after the
final audit. Killed: K. Ship-ready: F."
3. Do NOT post. The reader posts manually.

## On failure

- STOP the chain.
- Say exactly which stage failed and why.
- Do not regenerate or attempt a "best effort" pass.
- Suggest re-running the failed stage on its own.

## What this skill does NOT do

- Does not scrape data. The n8n weekly workflow does that.
- Does not draft X/Twitter posts. X is not part of this system.
- Does not generate giveaway resources. `giveaway-resource-generator` does that.
- Does not engage with comments. `linkedin-comment-engager` does that.
- Does not post to LinkedIn.

## Token efficiency

- No PDFs or reference files loaded at the orchestrator level.
- State passes between stages via file paths, not in-memory objects.
- One summary at the end, not a running commentary per stage.
```

post-curator STAGE 1

```
---
name: post-curator
description: Stage 1 of linkedin-content-pipeline. Reads the weekly watchlist scrape (watchlist-
posts.json) and filters it down to the posts that could actually help the reader build their
LinkedIn lead generation. Drops personal updates, motivational quotes, engagement bait, job posts,
and posts written for an unrelated audience. Keeps each surviving post with a short note on what
part of it is reusable as signal. Use when the pipeline invokes Stage 1 or the reader says "curate
the watchlist posts".
---

# post-curator (Stage 1)

Filters the raw weekly scrape into a clean set of posts worth learning from. The output feeds the
content-planner.

## Input

`watchlist-posts.json` — the raw weekly scrape from the n8n weekly content workflow. Every post from
every watchlist account, no filtering, with author, timestamp, post text, and engagement count per
```

post.

If the file is missing → STOP. "Run the weekly n8n content workflow first."

What to drop

Remove a post if it is any of these:

- A personal update with no transferable lesson (holidays, life news, anniversaries).
- A motivational quote or generic inspiration with no specific content.
- Engagement bait (one-line "agree?" posts, polls with no substance, comment-farming).
- A job post or hiring announcement.
- A pure promotion of the author's own paid offer with nothing useful in the body.
- A post written for an audience with nothing in common with someone trying to win clients on LinkedIn (for example, a post purely about a niche unrelated to LinkedIn growth, content, outreach, or client acquisition).

What to keep

Keep a post if it does any of these:

- Breaks down a workflow, system, tool, or process.
- Makes a specific, defensible claim about LinkedIn growth, content, outreach, or getting clients.
- Shows real numbers or a real result with enough detail to learn from.
- Takes a clear contrarian position with reasoning behind it.
- Names a real problem the reader would recognise.

How to judge engagement

Do not keep a post just because it has high raw engagement. Judge engagement **relative to the author's follower count**. A post from a 4,000-follower account pulling 60 reactions and a real comment thread is strong signal. A post from a 60,000-follower account pulling 40 reactions is weak signal for that account's size — the post underperformed. Weight the high-ratio posts as the stronger signal.

Output

`01-curated-posts.json`

For each surviving post, keep:
- The author and their approximate follower count (if available).
- The post text.
- The engagement count and a note on whether engagement was strong, average, or weak for that author's size.
- A one-line note: what is reusable here — a structure, a claim, a topic that is clearly landing this week.

Target output: 15-30 curated posts. If far fewer survive, say so — it usually means the watchlist is too thin or stale and needs more accounts added.

Hard rules

1. Curate, do not rewrite. This stage does not draft anything.
2. Drop generously. A smaller, cleaner set beats a large noisy one.
3. The source posts are signal, not material to copy. The note on each post says what it signals, never "rewrite this."

On failure

- `watchlist-posts.json` malformed or empty → STOP, do not proceed.
- Zero posts survive curation → STOP, tell the reader the watchlist needs more or better accounts.

content-planner

STAGE 2

```
---
name: content-planner
description: Stage 2 of linkedin-content-pipeline. Reads 01-curated-posts.json and produces the
week's plan — 7 LinkedIn post ideas, each with one single message, the signal from the curated posts
that points to it, and a research note. Enforces the content mix. An idea is a topic territory,
never a reworded version of one source post. Use when the pipeline invokes Stage 2 or the reader
says "plan the week's content".
---

# content-planner (Stage 2)

Turns the curated posts into a plan for the week: 7 LinkedIn post ideas, each ready for a writer to
draft.

## Input

`01-curated-posts.json` — the curated set from Stage 1.

If missing → STOP. "Run Stage 1 (post-curator) first."

## What an idea is

An idea is a topic territory with one clear message. It is **not** a rewrite of a single source
post. The curated posts are signal: they tell you what is getting traction this week, what problems
are being discussed, what is landing. The idea is built from that signal plus the reader's own angle
— never paraphrased from one post.

If two or three curated posts all circle the same topic, that is a strong signal the topic matters
this week — turn it into one idea, do not turn each post into its own idea.

## The 7-post mix (enforced)

Plan exactly 7 ideas to this mix:

- **3 ideas — LinkedIn lead generation.** How a system, workflow, or piece of automation brings
clients in. This is the hero theme. Vary the angle across the three: one could be a full workflow
walkthrough, one a single sharp tactic, one a before/after of how the reader's own approach changed.
- **2 ideas — buyer-language posts.** Name a real problem in the words the reader would use
themselves, then reframe it. The problem should come from the curated posts or from the recognisable
situations: posting rarely, posting without engagement, getting engagement but no clients, outreach
that stops when work gets busy, a pipeline that runs on referrals.
- **1 idea — a contrarian take** on LinkedIn growth or how people win clients. It must be
defensible, not contrarian for its own sake.
- **1 idea — a lead-magnet post** that ends with a comment CTA driving DMs. Only plan this if a
giveaway asset for it already exists. If no asset exists, replace it with a fourth lead-generation
idea and note that an asset needs building.

If 7 does not divide cleanly, favour the lead-generation theme.

## What each idea must carry

For each of the 7 ideas, write:
- **Type** — one of: lead-gen, buyer-language, contrarian, lead-magnet.
- **The single message** — one sentence. The one thing the post says. If you cannot say it in one
sentence, the idea is two ideas.
```

```
- **Signal** — which curated posts point to this topic, and what they signal (a topic landing this week, a problem being voiced, a structure that is working).
- **Research note** — what the writer needs to check or find before drafting: a current number, how a tool actually works, what is already overdone on this topic so the post avoids it.
- **Lead-gen bridge** — one line on what service or next step the post could naturally lead a reader toward. Every post should have a quiet bridge; none should be a hard pitch.

## Hard rules

1. **One message per idea.** No idea covers two things.
2. **Anti-paraphrase.** No idea is a reworded source post. If an idea reads like one curated post with synonyms swapped in, it is rejected and rebuilt from the topic up.
3. **The mix is fixed.** 3 lead-gen, 2 buyer-language, 1 contrarian, 1 lead-magnet. Do not drift.
4. **Plan 7, not more.** This is a one-week bundle.

## Output

`02-content-plan.json` — the 7 ideas, each with the fields above.

## On failure

- Fewer than enough curated posts to plan 7 ideas → STOP, tell the reader the watchlist needs more accounts.
- Cannot fill a mix slot honestly → say so rather than forcing a weak idea.
```

framework-picker

STAGE 3

```
---
name: framework-picker
description: Stage 3 of linkedin-content-pipeline. Reads 02-content-plan.json and assigns one writing framework to each of the 7 post ideas. The framework library lives inside this skill. A framework is picked on fit with the idea's shape, not for variety — if four ideas best fit the same framework, it is used four times. Use when the pipeline invokes Stage 3 or the reader says "pick frameworks".
---

# framework-picker (Stage 3)

Assigns one writing framework to each post idea, so the drafter writes into a fixed shape rather than inventing one. The framework library is below — this skill is self-contained.

## Input

`02-content-plan.json` — the 7 ideas from Stage 2.

If missing → STOP. "Run Stage 2 (content-planner) first."

## How to pick

For each idea, read its single message and pick the framework whose shape fits that message most naturally. Pick on fit alone. If the best fit for five ideas is the same framework, assign it five times. There is no rule against repeating a framework — forcing variety produces worse posts.

## The framework library

Each framework below is a shape, not a script. The drafter fills it with the reader's own message, research, and specifics. The frameworks are grouped by the job they do.
```

Group A — Showing how something works

These build credibility because they show real work.

The Build. For showing exactly how a system or process runs, step by step. The hardest shape for anyone else to copy.
Shape: open with the outcome the build produces. State how long it took to work out, or what it replaced. Then the steps in order, each step concrete enough to act on. Close with what the build now does without the reader's input.

The Teardown. For taking one thing apart — a workflow, a post, a tactic — and showing why each part is there.
Shape: name the thing. State why most people get it wrong. Walk each part in order, and for each part say what it does and what breaks without it. Close on the part that matters most.

The Before and After. For showing that the reader's own approach changed, without it reading as a list of wins.
Shape: how the reader used to do the thing, plainly. The moment or reason it stopped working. How the reader does it now. What changed as a result. The honesty of the "before" is what makes it land.

Group B — Naming a problem

These earn replies and saves from people who already half-feel the problem.

The Named Problem. For a problem the reader is living with right now, where you can name real fixes.
Shape: name the problem in the reader's own words, first line. State how long you have spent on it or how often you see it. Then the fixes, each one specific and real. Close on what changes once the fixes are in.

The Cost. For making an invisible problem visible by putting a number on it.
Shape: name something the reader treats as normal. Show what it is quietly costing — in time, in clients, in money. Make the number concrete and believable, not dramatic. Close on what that cost could be doing instead.

The Quiet Mistake. For a mistake the reader is probably making without knowing it.
Shape: name the mistake. Explain why it feels right, so the reader sees themselves in it without feeling attacked. Show what it actually causes. Give the correction. Close on how to tell if you are doing it.

Group C — Taking a position

These work when a topic is everywhere and you want to be the post that stands out.

The Contrarian Read. For a view most people in the space avoid saying out loud.
Shape: state the common belief. State your disagreement plainly. Give the reasoning — two or three points, each with something concrete behind it. Close on what to do instead. Only use this when the position can be defended in the comments.

The Reframe. For taking something the reader thinks is one kind of problem and showing it is another.
Shape: name what the reader thinks the problem is. Show why that framing keeps them stuck. Offer the truer framing. Show what becomes possible once the framing changes.

Group D — Driving an action

For the lead-magnet post only.

The Open Offer. For a post that ends by offering a free resource in exchange for a comment.
Shape: name a specific, real problem. Show that you have built something that addresses it — a document, a workflow, a checklist. Say briefly what is inside. Close with one clear instruction:

comment a single keyword and it gets sent. Never use this shape unless the resource already exists.

Output

`03-frameworks.json` — for each of the 7 ideas: the idea, the framework assigned, and one line on why that framework fits the idea's shape.

Hard rules

1. **One framework per idea.** Never two.
2. **Fit over variety.** Repeating a framework is fine and often correct.
3. **The Open Offer is only for the lead-magnet idea**, and only if the asset exists.
4. **Lock the framework before drafting.** Stage 4 writes into the assigned shape; it does not re-pick.

On failure

- An idea fits no framework cleanly → pick the closest and note the imperfect fit for the drafter, rather than forcing a wrong one.

linkedin-drafter STAGE 4

---
name: linkedin-drafter
description: Stage 4 of linkedin-content-pipeline. Reads 02-content-plan.json and 03-frameworks.json and writes the 7 LinkedIn posts. Each post is written in LinkedIn format from the start — short paragraphs, line breaks, a hook under 80 characters — using the assigned framework and Khyati's voice. Writes from the idea and research, never from a source post directly. Use when the pipeline invokes Stage 4.
---

linkedin-drafter (Stage 4)

Writes the 7 LinkedIn posts. One post per idea, each in the framework assigned at Stage 3.

Inputs

- `02-content-plan.json` — the 7 ideas, each with its single message, signal, research note, and lead-gen bridge.
- `03-frameworks.json` — the framework assigned to each idea.

If either is missing → STOP and say which.

Before drafting

For each idea, act on its research note. If the note says check a current number, how a tool works, or what is overdone on the topic — do that research first. Do not draft on assumption. A post built on a stale or wrong fact fails later stages.

How to draft

For each idea:
1. Take the single message. The whole post says this one thing.
2. Take the assigned framework's shape. Write into that shape.
3. Fill the shape with the reader's own angle, the research, and specifics. Never fill it by rewording a source post.
4. Format it as a LinkedIn post from the first line, not as an article to be reformatted later.

```
## LinkedIn format (built in, not added later)

- **Hook:** the first line. Under 80 characters. It states the actual point or tension, not a setup. It is the only line a reader sees before deciding to expand.
- **Paragraphs:** at most 3 lines on mobile. Break every 1-3 sentences.
- **Line breaks:** white space between thoughts. One idea per block.
- **Lists:** if the post lists things, use arrow bullets ( → ) not paragraphs of commas.
- **Length:** long enough to deliver the message, short enough that nothing is padding.
- **Close:** every post ends with the lead-gen bridge from the plan — a quiet next step, never a hard pitch. The lead-magnet post is the one exception: it ends with the comment-keyword CTA.

## Voice rules (enforced)

- First person ("I"). Never "we" — Khyati operates on her own.
- One thought per line. White space between thoughts.
- Plain language. Say the real thing without a setup sentence, a summary, or an announcement of what is coming.
- Full sentences with connector words. Not orphan fragments. "Not a discipline problem." on its own line is banned — write the full sentence.
- Reading level 6th-10th grade. Hard cap at 10th.
- No em dashes anywhere.
- No marketing pseudo-words: game-changer, leverage, unlock, 10x, supercharge, synergize, deep dive, circle back, moving forward, at the end of the day, in today's fast-paced world.
- Specific numbers, but no fake precision. "Around 25%" not "23.4%."
- No three-beat staccato rhythm (three short fragments in a row).
- No scenic or throat-clearing openings. The first line is the point.

## The credibility filter (enforced)

What the post may say:
- That Khyati builds AI systems and automations, and built her own LinkedIn system before building for anyone else. True and demonstrable.
- Confident framings on common observation: "most people I see...", "in this space...", "the people who get this right...".
- Workflows and systems Khyati could actually build if asked.
- Realistic, round, illustrative numbers in believable ranges.

What the post must never say:
- A named client, or a result delivered for a named client.
- A composite client story with named, specific details ("a consultant in London I worked with last year...").
- A specific revenue or metric outcome Khyati did not actually produce.
- Anything Khyati could not stand behind if a sales call asked for proof.

## The mechanism check (for lead-gen posts)

Any post that describes a specific workflow or system must pass this:
- The workflow could actually be built.
- The tools named exist and work as described.
- The steps are sequential and complete, with no hand-waving.

Buyer-language, contrarian, and lead-magnet posts do not need the mechanism check.

## Output

`04-drafts.json` — the 7 drafts, each tagged with its idea, type, and assigned framework.

## Hard rules

1. **One message per post.** If a draft drifts into a second topic, cut it back.
2. **Write from the idea, not the source post.** No paraphrase of watchlist posts.
```

3. **LinkedIn format from the first line.** Not an article.
4. **The credibility filter is absolute.** No named clients, no invented outcomes.

hook-editor

STAGE 5

```
---
name: hook-editor
description: Stage 5 of linkedin-content-pipeline. Reads 04-drafts.json and audits the first line of
every post. Scores each hook against a rubric, rewrites the weak ones, and confirms each hook's
promise matches what the post body delivers. Use when the pipeline invokes Stage 5 or the reader
says "fix the hooks".
---

# hook-editor (Stage 5)

Audits and fixes the first line of every draft. The hook gets its own stage because it is the line
that decides whether a post gets read at all.

## Input

`04-drafts.json` — the 7 drafts from Stage 4.

If missing → STOP. "Run Stage 4 (linkedin-drafter) first."

## What a hook has to do

The first line is the only part of a post a reader sees before deciding to expand it. A working
hook:
- States the actual point or the tension, not a setup for it.
- Is under 80 characters.
- Is specific enough that the right reader feels it is about them.
- Promises something the post body actually delivers.
- Has no AI cadence — no "Here's the thing", no rhetorical "Ever wondered...", no three-fragment
rhythm.
- Resolves its own pronouns. A hook that opens "He told me it wouldn't work" with no idea who "he"
is fails.
- Has no scenic opening. "It was a quiet Tuesday morning" is not a hook.

## How to audit

For each draft, score the first line against the points above. Mark it strong, weak, or failing.

- **Strong** — leave it.
- **Weak** — rewrite it. Keep the post's message; sharpen the line.
- **Failing** — rewrite it, and give three variant options for the reader to choose from.

## The promise-match check

Read the hook, then read the post body. If the hook promises one thing and the body delivers
another, the hook is failing even if it is well written. Fix the hook to match the body — do not
change the body to match a clever hook.

## Voice rules (same as the drafter)

- First person. No "we".
- No em dashes.
- No marketing pseudo-words.
- Plain language, no fragments, full sentences.
```

```
- Specific, no fake precision.

## Output

`05-hook-fixed.json` — the 7 drafts with hooks audited and fixed. For each, note: original hook,
audit result, final hook, and (if it was failing) the variant options offered.

## Hard rules

1. **Every hook is audited.** No draft skips this stage.
2. **Fix the hook to match the body**, never the reverse.
3. **Under 80 characters** is a hard limit.
```

post-reviewer

STAGE 6

```
---
name: post-reviewer
description: Stage 6 and final stage of linkedin-content-pipeline. Reads 05-hook-fixed.json and
audits every post on an 8-point evidence-required rubric. Auto-rewrites posts failing 3-5 checks.
Kills posts failing 6 or more, or any post that paraphrases a source post. Writes the ship-ready
bundle final-posts.md. The pipeline hard-stops here. Use when the pipeline invokes Stage 6 or the
reader says "review the posts".
---

# post-reviewer (Stage 6)

The last check before posts are ready. Audits every post, fixes what can be fixed, kills what
cannot, and writes the ship-ready file. The pipeline stops here. Nothing posts automatically.

## Inputs

- `05-hook-fixed.json` — the 7 drafts with hooks fixed.
- `01-curated-posts.json` — the curated source posts, used for the paraphrase check.

If either is missing → STOP and say which.

## The 8-point rubric

Audit every post against all 8. Each check needs a reason — a quoted line from the post and the rule
it satisfies or breaks. No check passes on a general impression.

1. **One message.** The post says one thing. If it covers two, it fails.
2. **Value.** A reader finishes the post with something they did not have before — a tactic, a
reframe, a number, a step. A post that only describes a problem with no payoff fails.
3. **Voice.** First person, plain language, no fragments, no em dashes, no marketing pseudo-words,
reading level at or below 10th grade.
4. **Hook.** First line under 80 characters, states the point, promise matches the body.
5. **Credibility.** No named clients, no invented client outcomes, no composite stories with named
details. Nothing the reader could not defend on a sales call.
6. **Mechanism (lead-gen posts only).** Any described workflow could actually be built, the tools
exist, the steps are complete.
7. **Format.** LinkedIn-native — short paragraphs, line breaks, arrow bullets for lists. Not an
article.
8. **Sounds human.** No AI cadence, no three-fragment rhythm, no scenic opening, no stating-the-
obvious warm-up, no purple prose.

## The paraphrase check (separate, and a hard kill)
```

```
For every post, compare it against the curated source posts. If the post closely mirrors the wording, structure, or specific phrasing of a source post, it fails the paraphrase check. This is not a scored check — it is a hard kill. A paraphrased post cannot be rewritten into a pass; it is removed and the slot is noted as needing a fresh idea.

## What to do with the scores

- **Passes all 8** — ship-ready, no change.
- **Fails 1-2 checks** — auto-rewrite to fix those checks, then re-audit.
- **Fails 3-5 checks** — auto-rewrite, but flag it for the reader to look at closely.
- **Fails 6+ checks** — kill it. Do not rewrite. Note the slot as needing a fresh idea.
- **Fails the paraphrase check** — kill it, regardless of the other 8.

## Expected kill rate

A healthy run kills 1-2 of the 7 posts. If a run kills nothing, the audit is being too soft and should be re-run more strictly. If a run kills 4 or more, Stage 2 or Stage 4 produced weak work and should be looked at. A bundle that ships fewer than 7 strong posts is better than one that ships 7 with weak ones in it.

## Output

- `06-final.json` — every post, its audit result, what was rewritten, what was killed.
- `final-posts.md` — the ship-ready bundle: the surviving posts, formatted exactly as they should appear on LinkedIn, ready to copy and paste. Each post labelled with its type (lead-gen, buyer-language, contrarian, lead-magnet) so the reader can pace the week.

## The hard stop

After writing `final-posts.md`, STOP. Show the reader:
1. The path to `final-posts.md`.
2. A one-line summary: posts that passed clean, posts rewritten, posts killed.
3. Do NOT post anything. The reader reviews and posts manually across the week.

## Hard rules

1. **Every check needs evidence.** A quoted line and the rule. No impressions.
2. **The paraphrase check is a hard kill.** No exceptions.
3. **Never post.** This skill writes a file and stops.
4. **A short clean bundle beats a full weak one.**
```

SECTION 05

The engagement skills

These two skills run outside the weekly content chain. The comment skill drafts what you say during the golden hour. The giveaway skill builds the resource a lead-magnet post hands out.

linkedin-comment-engager ENGAGEMENT

```
---
name: linkedin-comment-engager
description: Khyati Hooda's LinkedIn comment engagement agent for InboundLi. Reads the recent posts of people on the reader's engagement list and drafts 3 comment options per post in a plain, direct voice. Enforces a 2-engagement cap per person per rolling 30 days. When the list runs thin, suggests new people to add by looking at who engages with the people already on it. Use when the
```

reader says "draft today's comments", "run the comment engager", "engage on my list", or pastes LinkedIn post URLs.
---

linkedin-comment-engager

Drafts LinkedIn comments for the reader to post manually during the golden hour — the first 30-60 minutes after a post goes live, when commenting puts your name in front of the right people while the post is still being pushed.

First-time setup — the engagement list

This skill works from an **engagement list**: the people whose posts you want to comment on. These are people whose audience overlaps with the clients you want — so that when you comment well, the right people see your name.

The list starts empty. You fill it. The skill cannot guess who you want to engage with.

To build your first list, paste this to Claude:

> "Here are people I want to engage with on LinkedIn. Add them to my engagement list: [paste 10-20 LinkedIn profile URLs]."

Pick people who are active (posting in the last 30 days), whose posts get real engagement for their follower size, and whose audience is the kind of person you want as a client. Once the list has 15-20 people, the skill can run, and it will suggest more over time (see "Growing the list" below).

The list is stored at `engagement-list.json`. The skill reads it every run.

Inputs

- `engagement-list.json` — the people you want to engage with. Read every run. If it does not exist, this is the first run — tell the reader to build it using the setup prompt above, then stop.
- `prospect-posts.json` — the recent posts of the people on the list, scraped by the n8n daily comments workflow. If missing → STOP. "Run the daily n8n comments workflow first."
- `engagement-history.json` — the ledger of who has been engaged with and when. Read and written by this skill. If it does not exist, create an empty one and continue.

engagement-history.json schema

```json
{
  "last_updated": "2026-05-18T11:00:00Z",
  "profiles": [
    {
      "profile_url": "https://linkedin.com/in/example",
      "name": "Example Name",
      "engagements": [
        { "drafted_at": "2026-05-02T09:00:00Z", "post_url": "https://linkedin.com/posts/..." },
        { "drafted_at": "2026-05-11T09:00:00Z", "post_url": "https://linkedin.com/posts/..." }
      ],
      "engagement_count_30d": 2,
      "status": "capped",
      "cap_reset_at": "2026-06-01T09:00:00Z"
    }
  ]
}
```

The engagement cap

For each post in `prospect-posts.json`:
1. Look up the author in `engagement-history.json`.
2. Count engagements with `drafted_at` in the last 30 days.
3. If the count is 2 or more → SKIP this post. Log it to the skipped list.
4. If the count is 0 or 1 → eligible for drafting.

After drafting, add an entry to that person's `engagements`, recalculate `engagement_count_30d`, and if it reaches 2, set `status` to "capped" and `cap_reset_at` to 30 days after the earliest engagement.

The counter increments when a comment is **drafted**, not when it is posted. The reader posts most drafts; the occasional miss costs one slot, which is fine.

Drafting

For each eligible post:
1. Read the post.
2. Identify what kind of post it is:
   - **Tactical** — a workflow, tool, process, or breakdown.
   - **Opinion** — an observation, a take, commentary on the space.
   - **Win** — a result, a milestone, a piece of progress.
   - **Question** — the author asking their network something.
3. Draft 3 comment options, calibrated to the post type:
   - **Tactical:** add a specific point the author left out, or ask a sharp question about one step. Not a compliment.
   - **Opinion:** agree and add reasoning, or offer a different angle with reasoning. Framed as "different angle", never "you're wrong".
   - **Win:** specific acknowledgement that references a real detail in the post, plus one question about how. Not generic congratulations.
   - **Question:** answer with specifics, or offer an angle the author has not considered.

A good comment adds a point, shares an observation, or asks one real question. It never just paraphrases the post back at the author.

Voice rules

- First person ("I"). Never "we".
- Plain, direct, conversational — like reading a post and then talking to the person.
- 1-3 sentences. Short comments outperform long ones.
- No em dashes.
- No marketing pseudo-words (game-changer, leverage, unlock, 10x, deep dive, circle back).
- No casual filler (tbh, honestly, rn, lol).
- No emojis.
- No cheerleading openers — no "love this", "great post", "this is gold".
- Specific numbers where they help.

Growing the list

When the engagement list runs thin, or when a run has few eligible posts because most people are capped, suggest new people to add.

Find them the same way the reader built the first list: look at who consistently comments on the posts of people already on the list, and whose own posts get strong engagement for their follower size. Surface 5-10 suggestions with their profile URLs and a one-line reason each. The reader decides who to add — the skill never adds people on its own.

Tell the reader plainly: "Your list is getting thin / mostly capped. Here are people who keep showing up around the accounts you already follow — consider adding them."

Output

```
- `comment-drafts.md` — the drafts, grouped by how recently the post went live (last 30 minutes
first — comment now), 3 options per post, with the post excerpt and author. A capped-profiles list
at the top. If the list is thin, a "suggested additions" section at the bottom.
- `engagement-history.json` — updated with the new entries.

## Hard rules

1. **The engagement list starts empty and the reader fills it.** Never invent people for it.
2. **The 2-engagement cap is absolute.** A capped profile is skipped, no override.
3. **The counter increments on draft, not on post.**
4. **One comment per post.** Do not draft two comments on the same post.
5. **Never post.** The reader posts every comment manually.
6. **Suggest, never add.** New people are surfaced for the reader to approve.

## What this skill does NOT do

- Does not post comments to LinkedIn.
- Does not scrape posts — the n8n daily workflow does.
- Does not draft DMs.
- Does not add people to the engagement list on its own.

## On failure

- `prospect-posts.json` malformed → STOP.
- `engagement-history.json` corrupted → STOP, ask the reader to inspect it.
- All profiles capped, zero eligible posts → output the capped list and a set of suggested
additions, no drafts.
```

giveaway-resource-generator

ASSETS

```
---
name: giveaway-resource-generator
description: Khyati Hooda's giveaway resource generator for InboundLi. Creates the actual document
that gets sent by DM to people who comment on a lead-magnet post. Produces a practical, researched,
ready-to-share resource of roughly 2,000-4,000 words. Use when the reader says "create the resource
for [topic]", "build the giveaway asset", or "make the lead-magnet resource".
---

# giveaway-resource-generator

Creates the resource that a lead-magnet post promises. When someone comments the keyword on a lead-
magnet post, this is what gets sent to their DM.

## When to use

- "create the resource for [topic]"
- "build the giveaway asset for [post]"
- "make the lead-magnet resource"

## What it produces

A single document, roughly 2,000-4,000 words, that genuinely helps the reader with one specific
problem. It is practical and actionable — real steps, real tools, real workflow — not a strategy
overview. Someone should be able to act on it the day they receive it.

The resource is good enough that a reader thinks: if this is the free thing, the paid thing must be
serious.
```

```
## How to build it

1. **Take the topic** from the lead-magnet post the resource is paired with. The resource must deliver on exactly what the post promised — same problem, same audience.
2. **Research it.** Check what is current: how the relevant tools work now, what numbers are real, what has changed recently. Do not write from assumption.
3. **Structure it as a build, not an essay:**
   - A short opening — what this resource helps with and how to use it. No story.
   - The core — the actual practical content: steps, the workflow, the tools, what to do in order. This is most of the document.
   - A short close — one line on what to do if the reader wants this built and run for them, and how to reach Khyati.
4. **Make it usable.** Clear headings, steps in order, tools named, nothing hand-waved. If a reader follows it, it works.

## Voice rules

- First person ("I"). Never "we".
- Plain, direct language. Reading level at or below 10th grade.
- No em dashes.
- No marketing pseudo-words (game-changer, leverage, unlock, 10x, deep dive, supercharge).
- Specific numbers, no fake precision.
- About the reader, not about Khyati. Minimal intro, minimal outro.

## The credibility filter

- No named clients, no client result claims.
- No composite stories with named details.
- Realistic, round, illustrative numbers only.
- Everything in the resource is something Khyati could stand behind on a sales call.

## No overlap

If the resource covers LinkedIn rules, frameworks, or growth tactics, it must not mirror the naming or structure of any third-party guide. Write it fresh, in Khyati's own framing.

## Output

A single markdown document, ready to convert to PDF or share as-is. Named clearly for the topic it covers.

## Hard rules

1. **It must deliver on the exact promise** of the lead-magnet post it pairs with.
2. **Research before writing.** No assumption-based content.
3. **Practical, not strategic.** Real steps a reader can act on.
4. **The credibility filter is absolute.**
```

SECTION 06

The daily engagement workflow

The comment skill needs source material the same way the content chain does. A second n8n workflow provides it. It runs daily, reads your engagement list, pulls the recent posts of a few people on it, and saves them for the comment skill to read.

The flow

```
Manual trigger
  -> Read engagement list sheet
  -> Pick 5 people (5-day cooldown, 2-per-30-days cap)
  -> Scrape recent posts via Apify (one person at a time)
  -> Combine into one file
  -> Save dated JSON to Google Drive
  -> Update the sheet with engagement tracking
```

How it differs from the weekly workflow

The weekly workflow scrapes every account on the watchlist, because the content chain wants a wide pool of source material. The daily workflow picks only five people, because you can only meaningfully comment on a handful in a day, and because it enforces a cap: nobody gets engaged with more than twice in a rolling 30 days. Engaging more than that reads as following someone around, not as genuine engagement.

> **The engagement list starts empty too.** Build it the same way as the watchlist: people whose audience overlaps with the clients you want, who post regularly, with real engagement for their size. The comment skill will suggest more people to add over time, by looking at who consistently shows up around the accounts already on your list. It suggests; you decide.

Both workflows ship as importable files alongside this document. Import them into n8n rather than rebuilding by hand. Each one has a small set of placeholders written in capitals to replace after import: your sheet IDs, your Apify token, your Google credentials, and your Drive folder IDs.

SECTION 07 The tools you need

A one-person operation can run all of this. Here is the full list and what each part is for.

TOOL	WHAT IT DOES
Claude	Runs every skill in the chain. Used with the file system on, so skills can read and write the files they pass between each other.
n8n	Runs the two scraping workflows, weekly and daily. Self-hosted or cloud-hosted.
Apify	Does the LinkedIn scraping, through the apimaestro/linkedin-profile-posts actor. Priced per run; a weekly watchlist scrape and a daily engagement scrape cost very little.

Sales Navigator	Used to find the people who go on your watchlist and your engagement list, and to find prospects to reach out to. LinkedIn's own search, with filtering depth the free search does not have.
Google Sheets	Holds the watchlist and the engagement list. The only part you edit by hand.
Google Drive	Holds the files passed between n8n and Claude.

Roughly what it costs to run

The two paid pieces most people do not already have are Sales Navigator and Apify. Sales Navigator is the larger of the two. Apify is usage-based and small at this volume. n8n has a free self-hosted option. Claude you likely already have. The whole stack runs for a modest monthly figure, far less than a single client is worth.

> **The honest part.** The tools are the cheap part. The real cost is time. Building the watchlist properly, finding the right people for the engagement list, tuning the skills to sound like you, and holding the daily engagement habit all take hours, and they take judgement. The system removes the work that does not need your judgement. It does not remove the work that does.

SECTION 08 How the pieces connect

The full loop, from a watchlist to a posted post.

```
Watchlist sheet
  -> n8n weekly run -> Apify scrape -> dated JSON in Google Drive
  -> Claude skill chain (Stage 1 through Stage 6)
  -> final-posts.md, a bundle of 7 finished posts
  -> your review
  -> you post each one across the week

Engagement list sheet
  -> n8n daily run -> Apify scrape -> dated JSON in Google Drive
  -> linkedin-comment-engager
  -> comment-drafts.md, comments for the golden hour
  -> your review
  -> you post each comment manually
```

The order to build it in

1. The weekly workflow and the watchlist. This gives you a steady supply of source material, which is the part most people never set up.

2. The skill chain. The orchestrator and the six stages. Add them one at a time; each stage is useful on its own.

3. The daily workflow and the comment skill. Once you are posting consistently, add the engagement layer.

4. The giveaway skill. Build it when your first lead-magnet post needs a resource to hand out.

What stays manual, and why

The review at the end of a run and the posting itself stay with you. So does holding the daily engagement habit. The system is built so the work it does is the work that does not need your judgement, and the work you keep is the work that does. That line is the whole design.

> **If you would rather not build it.** This is the complete system and everything you need to build it is in this document. It also takes real time, real tools, and steady attention. If you would rather have it built and run for you, so the leads come in without you assembling any of this, that is what I do. Message me on LinkedIn and I will show you how it would work for you.
>
> **Khyati Hooda** · linkedin.com/in/khyatihooda · inboundli.com