Superlinked

STARTER KIT · SUPERLINKED

Bring AI in-house

How to take a job a person does today, hand its narrow steps to small models, and run those models where the data lives.

STEP 1 · THE JOB TO BE DONE

Start from one job, and draw the workflow a person follows today

Pick a job someone already does every week and write down each step. Here is a claims processor checking a repair invoice against its claim.

1 · THE INVOICE ARRIVES
A scan or a phone photo

↓

2 · READ IT
The shop, the claim number, the lines, the total

↓

3 · OPEN THE CLAIM
The insured, the vendor, the authorized amount

↓

4 · COMPARE THE TWO
Right claim, right shop, lines add up, total within what was authorized

↓

5 · DECIDE
Pay it, or question it

> Most invoices are fine. That’s the trouble: after the fortieth clean one, the forty-first gets the same quick look.

STEP 2 · SMALL MODELS

Mark the steps a small model you run can take

For each step, start from the cheapest owner: code for a lookup, a sum or a rule, then a small task model for reading, then a large model, then a person. On the invoice check it comes out like this.

Read the scan	→	**SMALL MODEL** OCR, with LightOnOCR
Find the claim number and the total	→	**SMALL MODEL** Extraction, with GLiNER2
Open the claim	→	**LOOKUP** The claims system already knows it
Compare the two	→	**RULES IN PYTHON** No model in it
Pay it, or question it	→	**A PERSON** The claim handler approves

> Two of the five steps needed a model, and both are small ones. The rest was already in your systems or was a rule you could write.

Let the steps decide between a workflow and an agent

The steps are known beforehand You can write them down before you see the case	→	**WORKFLOW** Your code runs each step on an event or a schedule, and small models do the reading.
The steps change completely with what you see	→	**AGENT** A model, usually a large one, works out the steps, and code checks every part of the result it can.
The steps change only in some parts	→	**HYBRID** The workflow runs as usual and hands what it can't settle to an agent.

The invoice check is a workflow. A few times a month the fraud team asks about one repair shop's whole year, and nobody can write those steps down, so an agent runs that job and calls the invoice check as a tool. It added about $0.09 a job.

> A person decides at the end in all three. A workflow knows its cost per item up front, while an agent pays for the model that runs its loop on every turn.

Cookbook recipe and its runs: insurance/check_invoice_against_claim →

STEP 4 · CHECKING EACH OUTPUT

Check what the model says against data it didn’t produce

INDEPENDENT CHECKS

- The claim in your claims system
- The purchase order in your ERP
- Is the span really in the source text?
- For an agent, what its tools returned

USEFUL, BUT NOT INDEPENDENT

- Arithmetic over numbers the model extracted: it catches a misread line, but a made-up value that balances passes
- A model’s own score against its own threshold
- An agent’s own account of what it did

On one invoice the OCR model wrote “$76.20” into a blank tax line, exactly 6% of the subtotal. The arithmetic check passed, because it only compared the model’s numbers with each other.

> So feed a model only the text it should judge, and keep a person on whatever code can’t settle.

Cookbook recipe and its runs: insurance/check_invoice_against_claim →

STEP 5 · EVALUATION

Test it on your own data before you trust it

A WORKFLOW	AN AGENT
Runs the same steps at volume with nobody watching. Test the inputs you expect, plus the edge cases.	Takes a different route each run. Test many different requests, odd ones included, and run each one more than once.

We ran each agent job in the cookbook five times. On a bank’s look-back over the accounts it opened in 2025, code checked the agent’s answer against the account register:

5	5 / 5	5 / 5
runs of each agent job	missed the same account	sent back by the check

> Five out of five is a pattern you can fix. Score cutoffs need the same care: recheck one when the model or the hardware changes, and send scores near it to a person.

Cookbook recipe and its runs:
finance/check_kyc_documents_against_policy →

STEP 6 · OFF THE SHELF OR FINE-TUNED

Fine-tune when the model makes the call and falls short on your data

0.20	0.62	0.84
GLiNER2.5-Decide off the shelf	Hosted decision API, given the annotation guidelines	ModernBERT-large fine-tuned on 5,532 clauses

F1 on the unfair class. Superlinked's study on LexGLUE UNFAIR-ToS: 3,882 lawyer-labeled Terms of Service clauses scored, 402 of them unfair; the fine-tune trained on 5,532 others. Study · Method

When the model only points at something and code makes the call, use it off the shelf with rules around it. When the call is the model's own judgment, like "is this clause unfair?", measure the model as it ships on your labeled data, and fine-tune if it falls short.

> You probably have the labels already, so train a classical classifier on them first, and fine-tune only if it misses your target. SIE serves models and doesn't train them; SIE Cluster or SIE Local serves what you trained.

STEP 7 · BRING IT IN-HOUSE

Decide where the models run from the workload

SIE SERVERLESS	SIE CLUSTER	SIE LOCAL
Superlinked runs the GPUs. Inputs go to Superlinked.	Your cloud, your GPUs. Inputs stay in your cloud.	Your machine. Inputs stay on it.

- [ ] **Have you picked the models yet?**
  If not, try them on SIE Serverless; there's nothing to size a GPU for.
- [ ] **What do your data rules and infra team allow?**
  Data that can't leave your cloud means SIE Cluster or SIE Local. An agent's loop on a frontier model sends its inputs out either way.
- [ ] **How spiky is the traffic?**
  A few runs a day fits SIE Serverless; a GPU busy most of the day favors SIE Cluster.
- [ ] **Where does the data sit?**
  On one workstation, SIE Local. Flowing through production, SIE Cluster.

Choosing between SIE Serverless, SIE Cluster and SIE Local →

WHERE SIE FITS

Your code stays where it is, and only the models move

YOUR MACHINES

YOUR APP OR WORKFLOW
Python does the lookups, the sums and every rule check.

OPTIONAL: AN AGENT LOOP
Often a large model plans; its tools call the small models.

↓

ONE API FOR EVERY TASK MODEL · SIE IS APACHE 2.0

SIE SERVERLESS
Superlinked's GPUs.

SIE CLUSTER
Your cloud.

SIE LOCAL
Your machine.

↓

CHECK EACH MODEL IS OFFERED WHERE YOU RUN IT

TASK-SPECIFIC OPEN MODELS

- OCR: LightOnOCR
- Document parsing: Docling
- Extract: GLiNER2
- Classify: GLiNER2.5-Decide
- Redact PII: pii-redact
- Rerank: Qwen3 Reranker
- Speech to text: parakeet

```
# SIE Serverless
SIE_API_KEY=<your key>
# SIE Local, or https://<your-gateway> for SIE Cluster
SIE_BASE_URL=http://localhost:8080
```

App code stays the same; server setup is separate. 29 of 46 recipes need several images, which SIE Cluster's gateway routes behind one URL. Qwen3-Embedding-4B has no self-hosted image, so recipes use the 8B. Self-hosting notes →

CHECKLIST

What to check before you bring a job in-house

- [ ] **Write the job down, one step at a time.**
  What a person does today, with each step's input and output.
- [ ] **Give each step the cheapest owner that works.**
  Code, a small model, a large model, a person.
- [ ] **Decide between a workflow, an agent and a hybrid.**
  Steps you can write down make a workflow.
- [ ] **Name the data each check compares against.**
  It should come from your systems, never from the model.
- [ ] **Test on your own data, and run agent jobs more than once.**
- [ ] **Fine-tune only when the model makes the call and falls short on your data.**
  On your labeled data, and after a classical classifier misses the target.
- [ ] **Decide where the models run from the workload.**
  Start with IT security: where may the data go?
- [ ] **Name who works the review queue, and why it's them.**

THE COOKBOOK

We made each of these decisions in the cookbook

About 50 recipes in nine industries, workflows and agents, with a post per industry on how we divided the work, wrong turns included.

The cookbook on GitHub
github.com/superlinked/cookbook →

Get an SIE API key
console.superlinked.com →

THE POSTS BY INDUSTRY

- E-commerce →
- Retail →
- Insurance →
- Healthcare →
- Finance →
- Legal →
- Manufacturing →
- Security →
- SRE →

AND THE ONES ACROSS INDUSTRIES

- How companies use small, open-weight and task-specific models →
- When to use a workflow and when to use an agent →
- Give your agent small open models as tools →
- Choosing where small models run →
- Who owns the tokens? →

SCAN FOR THE COOKBOOK