

THE PLAYBOOK

Build your own AI agent stack

A crew of small, single-job AI agents that quietly run your operations — brief you each morning, prep your meetings, build and ship your client work, run your content — with a human still approving every real move. This is the exact way I build mine.

BY CORINA • THE CV METHOD • @THE_CV_METHOD

WHAT YOU'RE BUILDING

Forget the one do-everything mega-bot. The trick is the opposite: many narrow agents, each with a single clear job, tight guardrails, and a human who ships. Small agents are easier to trust, test, and fix — and together they add up to a system that runs the business. You don't launch the whole stack at once; you grow it one reliable agent at a time.

RULE ZERO

One job per agent — and a human still ships. It proposes, you approve.

A narrow job is testable and safe; the human gate is the single thing that stops an agent from doing the wrong thing, confidently, at scale.

01 List the repeatable tasks bleeding your time

The morning brief, the follow-ups you forget, the deploy, the content draft, the monthly report. Every task that repeats and has a clear "done" is a candidate for its own agent. Write them down, then pick the one costing you the most time this week — that's agent #1.

If you can't say the job in one sentence, it's too big — split it into two agents.

Boring and bulletproof beats shiny. Here's what I actually use — and the general equivalent so you can copy the shape without locking in.

PIECE	WHAT IT DOES	I USE	SWAP FOR
Agent runtime	where the agents actually run & use tools	Claude Code or the Claude desktop app	any capable AI agent runtime
A home + memory	one place agents read your world	an Obsidian vault + a CLAUDE.md system doc	any notes app + a "system" doc
Where work ships	the apps/services agents act on	Cloudflare Workers / Pages	any serverless, or the tools you already pay for
A registry	a list of what each agent does & when it fires	a small agents.js + a table	a spreadsheet or a single doc
A schedule	runs the daily agents on their own	scheduled tasks (weekday routines)	any scheduler / cron

03

GIVE EACH AGENT ITS CONTEXT · MEMORY · FEED IT YOUR WORLD

A home folder + a short "system" doc

An agent is only as good as what it knows. Give it a folder and a short doc (mine is a CLAUDE.md) that spells out your world: what the project is, the stack, the commands, and the always / never rules. Feed it the real context once so it stops guessing on every run.

Write the never-rules down explicitly: "never deploy without a security pass," "never send an email — only draft." The doc is the guardrail.

04

WRITE THE AGENT BRIEF · THE JOB · ONE PAGE

Job, inputs, "done," and guardrails

For each agent write a tiny brief: its one job, the inputs it gets, the output / definition of done, and its guardrails. This is the whole difference between "vibes" and an agent you can rely on. Keep it short and specific — a paragraph, not a manifesto.

Add a test: two example inputs and the right answers. That's an "eval" in plain terms — you can't trust what you never checked.

05

ADD THE SAFETY GATE · RULE ZERO, MADE REAL

Nothing ships without a check + your yes

Put a gate between "the agent did the work" and "it goes live": a security / QA pass that returns PASS or BLOCK and fails closed (if in doubt, it blocks). The agent proposes; a human approves. This one habit is what lets you trust the rest of the stack.

Make the gate boring and automatic — the same checklist every build has to clear, no exceptions, no "just this once."

06 Daily agents run themselves; the rest on demand

Put the everyday agents (the brief, the end-of-day capture) on a schedule so they just happen; invoke the others when you need them. Keep a simple registry of what each agent does and when it fires, and review the outputs — tune the briefs as you learn. Start with one agent, make it reliable, then add the next.

THE RULE OF GROWTH

A stack is grown, not launched. One trustworthy agent beats ten half-built ones.

A • YOUR SYSTEM DOC – THE CONTEXT EVERY AGENT READS (A CLAUDE.MD)

```
# <Your business> – operating doc

## What this is
<one line: what the business does>

## The stack
<the tools the agents work with – your CRM, dashboards, where work ships>

### Always
- propose, then wait for my approval before anything ships / sends / posts
- keep client work anonymized (category labels, never real names)
- write back what changed when you finish

## Never
- deploy without a PASS from the security gate
- send an email or post on your own – draft only
- invent a number or a source

## The agents
<one line each: name – its single job>
```

B • THE AGENT BRIEF – ONE PER AGENT

```
# Agent: <name>
Job:      <the ONE thing it does>
Trigger:  "<the phrase that runs it>"
Inputs:   <what it starts with>
Steps:
  1. <first step>
  2. <next step>
Done =    <exactly what a finished result looks like>
Guardrails:
  - never <the hard no for this agent>
  - always a human approves before it ships
Test:     given <example input>, it should <expected output>
```

C • THE PRE-SHIP SECURITY GATE — THE CHECKLIST IT MUST CLEAR

Before anything goes live, run the gate → return PASS or BLOCK (fail closed):

- ☐ Secrets / keys are server-side — never in code or prompts
- ☐ Access is scoped to only what this agent needs
- ☐ No private or client data would ship or leak
- ☐ Inputs are validated; errors fail safe
- ☐ A human has approved the go-live

BLOCK if any box is unchecked. If in doubt, BLOCK.

D • SPIN UP YOUR FIRST AGENT (PASTE THIS TO SCAFFOLD IT)

Help me stand up one AI agent, safely.
The job I want to hand off first: <the one repetitive task>.

Give me:

1. a one-paragraph agent brief (job, inputs, "done", guardrails)
2. the always / never rules to add to my system doc
3. two test cases with the right answers
4. where a human should approve before anything ships

Keep it narrow — one job — and assume it proposes, I approve.

GET THE BUILDING BLOCKS

github.com/anthropics/skills — the skill-creator (it scaffolds an agent for you), a skill template, and the Agent Skills spec. In Claude Code: /plugin marketplace add anthropics/skills.

DON'T PULL, BUILD

unlike off-the-shelf skills, your stack is yours — narrow jobs specific to your business. Use the scaffold prompt above (D), or the skill-creator, to stand up the first one.

RUN THEM

each agent is a skill in your `.claude/skills/` folder, run by Claude Code — put the daily ones on a schedule so they fire on their own. On Claude.ai, add them under Customize → Skills.

DOCS

support.claude.com → "creating custom skills" · the open standard: agentskills.io

KEEP IT HONEST & SAFE

You approve before anything ships

No invented metrics

Real work stays anonymized

Responsible data practices

Keep secrets out of prompts & anything you publish

THE STARTER CHECKLIST

- ☐ One job written down for agent #1
- ☐ A brief: job · inputs · done · guardrails
- ☐ A security / QA gate that fails closed
- ☐ A registry of what each agent does
- ☐ A home folder + a system doc (your world, always/never rules)
- ☐ A test case or two (your "eval")
- ☐ The daily agents on a schedule
- ☐ A human approves every live / irreversible action

Want this built for you instead?

I install the whole crew — briefs, guardrails, the security gate, and the schedule — wired into the tools you already use, and hand it over documented. That's the CV Method.

hello.thecvmethod.com →



Strategy that ships.

hello.thecvmethod.com