THE EXECUTION GUIDE

Build with *Claude.*

Five systems. One weekend. Hours back every week.

Read this first

This guide turns the Build with Claude playbook into something you can actually execute. Five systems. Each one ships in a weekend. Each one buys you back hours per week.

What you'll build

1. Custom Claude Skills — programmable workflows in markdown that train Claude to do specific jobs in your voice.
2. MCP Connections — Claude wired into the 100+ tools you already use (Gmail, Notion, GitHub, Stripe, etc.).
3. Internal Knowledge Assistant — your company's brain inside Claude, ready to answer team and customer questions.
4. Multi-Agent System — an AI assembly line where multiple Claude agents handoff work, supervised by one human (you).
5. Memory + Workflow Chains — business intelligence Claude actually retains, and multi-step workflows that pass context cleanly.

How to use this guide

- Read each section once, start to finish, before executing.
- Pick ONE system to build this week. Don't start the next until the first one ships.
- Follow the steps in order. Don't skip — they're sequenced for a reason.
- Default to the simplest option. You can always optimize later.
- Iterate based on what actually breaks, not what you imagine might break.

> **TIME BUDGET** First system: 1–2 hours. Full stack: one weekend. Hours saved after: 30+ per week.

Prerequisites

- A Claude.ai account (Pro or Team tier, so you can create Projects).
- A Claude API key from console.anthropic.com (for sections 3, 4, 5).
- An n8n account — free tier is fine — for the multi-agent flows.
- A Pinecone account — free tier — for shared agent memory.
- Your help docs, SOPs, and product documentation in one Drive folder.

Section 1 — Skills (.md files)

> **WHAT YOU SHIP** A skill file Claude auto-loads on every relevant chat. Build five, replace 80% of your daily prompts forever.

What a Skill actually is

A Skill is a markdown file (SKILL.md) that tells Claude how to do a specific job. It's not a prompt — it's a persistent capability you upload once, and Claude auto-loads it whenever the trigger pattern appears.

Think of it as hiring an employee with a custom handbook. The skill defines tone, format, rules, and reference examples. Once uploaded, every output respects it.

Why this matters

Most users retype the same prompt every single chat. Skills replace that. Build five skills once, and Claude knows your voice, formats, and rules forever. The leverage compounds with every use.

Step-by-step: Build your first Skill

Step 1 — Pick your first skill

Start with writing-voice.md. It's the highest-leverage skill because every written output benefits from it. After that, build lead-research, then carousel-builder, then workflow-viz, then report-engine.

Step 2 — Create the file

Open any text editor (VS Code, TextEdit, Notepad). Save a new file as writing-voice.md.

Step 3 — Add the skill header (YAML frontmatter)

At the top of the file, between three dashes:

```
---
name: writing-voice
description: Apply Vedant's tone and voice rules to any written output.
---
```

Step 4 — Define your voice rules

Use this template (fill in your specifics):

```
# Voice Rules

## Tone
- Conversational but sharp. Confident, not arrogant.
- Direct. Never hedge with "I think maybe" — make the claim.
```

```
## Banned words
Never use: "leverage", "unleash", "synergy", "in today's fast-paced",
"game-changer", "revolutionary".

## Required moves
- Start with a hook. Never with "In this post" or "Today we're going to".
- Short sentences. Then a longer one to pull the reader through.
- End with one specific action the reader can take in 60 seconds.

## Format defaults
- Sentence case headlines, not Title Case
- Em-dashes for asides — like this — never parentheses
- Italics for emphasis. Never bold for emphasis.

## Voice examples
[Paste 3–5 paragraphs of your best published writing here. Claude
mimics the cadence from these examples more than from rules.]
```

Step 5 — Upload to Claude

6. Open claude.ai in your browser.
7. Click Projects in the left sidebar, then New Project.
8. Name it: "Your Personal Stack".
9. Click the Project knowledge tab inside the project.
10. Upload writing-voice.md.

Step 6 — Test the skill

Open a new chat inside that Project. Try:

```
Write a 200-word LinkedIn post about pricing strategy, using my
writing-voice skill.
```

The output should sound like you. If it doesn't, your examples need to be more representative — go back to Step 4 and paste 3 more of your best pieces.

> **PRO TIP** Skills are auto-loaded by Claude based on relevance — but for the first few uses, mention them explicitly in your prompt to make sure they activate.

The five skills to build (in order)

1. writing-voice.md

- What it does: Auto-applies your tone, banned words, signature phrases to every written output.
- Time to build: 30 minutes.
- Where it pays off: LinkedIn posts, emails, scripts, captions — every text output.

2. lead-research.md

Drop a name into the chat, Claude returns a structured dossier with hooks for outreach. Template:

```
---
name: lead-research
description: Research any lead and return a structured outreach dossier.
---

# Lead Research Skill

## Input
A person's name (+ optional company).

## Output structure
1. Name, role, company
2. Location and background
3. Recent news (last 3 months — actual events, not boilerplate)
4. Their content themes (top 3 from their posts/articles)
5. Three personalized outreach hooks — one sentence each + why
   it would land with this specific person.

## Sources to check
- LinkedIn profile + recent posts
- Twitter/X handle
- Their company's blog
- Recent press mentions or podcast appearances

## Rules
- Cite every fact with a source.
- Skip generic info (industry trends, common knowledge).
- Each hook references something specific to them.
```

3. carousel-builder.md

- What it does: Takes a topic + format preference, outputs a 10-slide carousel with hook, body slides, and CTA, all in your voice.
- How: Reference your writing-voice skill inside this skill, then add carousel-specific format rules.
- Pays off: Every time you make a carousel, this skill does 70% of the work.

4. workflow-viz.md

- What it does: Describe any process in plain language, get back a diagram + flowchart (text-based, ready to render in Mermaid or similar).
- Great for: System design, onboarding flows, agent architecture, documentation.

5. report-engine.md

- What it does: Takes raw data + your template, outputs a polished report. Pulls from Drive Connector if available.
- Use it for: Weekly reports, client updates, internal status, investor memos.

Troubleshooting

- Skill not loading? Verify it's in Project knowledge — not just pasted into a chat.
- Output drifting from voice? Add more examples to Step 4. Examples beat rules every time.
- Skill being ignored? Reference it explicitly: "Use my writing-voice skill."
- Skill triggering when you don't want it? Make the description more specific (so it only auto-loads for true matches).

Section 2 — MCP connections

> **WHAT YOU SHIP** Five connectors plugged into Claude — Firecrawl, Brave Search, GitHub, Notion, Stripe. Claude stops being a chatbot and becomes an OS on top of your stack.

What MCP actually is

Model Context Protocol — an open standard from Anthropic that lets Claude connect to external tools and data sources. Instead of building API integrations from scratch, you install an MCP server. Each server exposes its tool's capabilities to Claude.

In plain terms: MCP is how Claude talks to Gmail, Notion, GitHub, your database, your CRM. One install, full access.

Step-by-step: Install your first MCP (Firecrawl)

Step 1 — Get a Firecrawl account + API key

11. Go to firecrawl.dev.
12. Sign up. Free tier gives you 500 scrapes per month — plenty to start.
13. Dashboard → API Keys → Copy your key. Save it somewhere safe.

Step 2 — Connect Firecrawl to Claude

14. Open claude.ai.
15. Click your profile icon → Settings → Connectors.
16. Search the directory for Firecrawl. Click Connect.
17. Paste your API key. Authorize.

Step 3 — Test it

Open a new chat and try:

```
Use Firecrawl to scrape https://news.ycombinator.com and summarize
the top 5 stories.
```

Claude will fetch the page, parse it cleanly, and respond with summaries. Compare to manually scrolling HN — you just saved 10 minutes.

> **IF FIRECRAWL ISN'T IN THE DIRECTORY** Some MCPs are added manually as custom connectors. In Settings → Connectors → Custom MCP, paste the MCP URL (usually in the tool's docs).

The 5 famous MCPs to install today

1. Firecrawl — Web scraping

- Sign up at: firecrawl.dev
- Setup: Settings → Connectors → Firecrawl → paste API key.
- Test prompt: "Scrape [URL] and extract the main content as markdown."
- Pays off in: Research, competitor analysis, content monitoring.

2. Brave Search — Live web search

- Sign up at: brave.com/search/api
- Free tier: 2,000 queries per month.
- Setup: Get API key from Brave dashboard → Settings → Connectors → Brave Search.
- Test prompt: "Search Brave for the latest AI funding news this week."
- Pays off in: Anything that needs live, current information.

3. GitHub — Code agent

- No API key needed — OAuth flow.
- Setup: Settings → Connectors → GitHub → Authorize → pick which repos Claude can access.
- Test prompt: "List my last 5 PRs and summarize what changed in each."
- Pays off in: Code review, changelog generation, PR descriptions, repo audits.

4. Notion — Workspace sync

- No API key — OAuth.
- Setup: Settings → Connectors → Notion → Authorize → select pages/databases to share.
- Test prompt: "Find my notes on Q3 strategy and summarize the latest entries."
- Pays off in: Internal knowledge, project tracking, content libraries.

5. Stripe — Customer + payment data

- Setup: Stripe Dashboard → API keys → create a Restricted Key (read-only is fine for most uses).
- Connect: Settings → Connectors → Stripe → paste restricted key.
- Test prompt: "Show my top 10 customers by lifetime value with their current plan."
- Pays off in: Customer success workflows, churn analysis, billing questions.

Chaining MCPs (where it gets serious)

Once you have 3+ MCPs connected, real workflows unlock from single prompts:

```
Search Brave for our top 3 competitors → scrape their pricing pages
with Firecrawl → summarize the differences → save the summary to my
Notion competitor-intel page.
```

That's a one-prompt market intelligence pipeline. Used to be a 2-hour task.

> **PRO TIP** Start with 2 MCPs (Firecrawl + Brave). Get those workflows clean before adding more. Too many connectors at once = Claude getting confused about which tool to use.

Section 3 — Internal knowledge assistant

> **WHAT YOU SHIP** A Claude-powered assistant that answers team questions from your actual docs, drafts customer support replies, and surfaces ticket patterns. Your business brain, on call 24/7.

What you're building

Internally: team members ask "how do we do X?" and get cite-backed answers from your SOPs and docs. No more pinging the same senior person for the same questions.

Externally: customer support tickets get drafted replies pulling from your help docs and past tickets. A human approves and ships in seconds.

Step-by-step

Step 1 — Collect your knowledge sources

Centralize everything in one Google Drive folder. Pull together:

- SOPs (Standard Operating Procedures) — as Docs or PDFs
- Product documentation — user guides, API docs, feature explainers
- FAQ database — export from Intercom, HelpScout, Zendesk, or wherever it lives
- Internal wikis — export Notion as markdown, or just share the relevant pages
- Past resolved tickets — a sample of 50–100 well-handled tickets is gold

Step 2 — Create the Claude Project

18. Claude.ai → Projects → New Project.
19. Name: **Internal Assistant** (or your team's name)
20. Add a Project Description explaining what this assistant does and who uses it.
21. Upload every doc from Step 1 to Project Knowledge. You can drag-and-drop dozens at once.

Step 3 — Add system instructions

In the Project's Instructions field, paste this template (customize the bracketed parts):

```
You are [Company]'s internal assistant. Your job:

1. Answer team member questions using only our internal documentation.
   Always cite the source doc.

2. Draft customer support replies in our tone: warm, direct, no jargon.
   Format as ready-to-send drafts.

3. When asked, surface patterns in our support tickets — common
```

```
   issues, sentiment shifts, feature requests.

Rules:
- Cite the source doc for every claim. No source = say "I couldn't find
  this in our docs — escalate to [name]."
- Default to brief answers. Expand only when asked for detail.
- Never speculate about pricing, contracts, or legal questions —
  flag those for a human and stop.
- Match our voice: confident, plain English, no corporate-speak.
```

Step 4 — Test with real questions

Run a few test queries you know the docs cover:

- "How do we onboard enterprise SaaS clients?"
- "What's our refund policy for annual subscriptions?"
- "Draft a reply to a customer asking why feature X isn't working."

Check every answer against the source docs. If it drifts, refine the Project Instructions or add more docs.

Step 5 — Connect to Slack (optional but huge)

Make the assistant available to your team where they already work:

Option A — Slack Connector (simplest)

22. Settings → Connectors → Slack → Authorize your workspace.
23. In Slack, tag @Claude in any channel with your question.
24. Claude responds in the thread, pulling from Project knowledge.

Option B — Bridge via Zapier/Make (more control)

25. Zapier trigger: New message in a specific Slack channel (e.g., #ask-support).
26. Action: Claude API call with the message + project context.
27. Response: Post Claude's reply back to the Slack thread.

Step 6 — Connect to your helpdesk (optional)

For customer support automation:

28. Set up a webhook from your helpdesk (Intercom/Zendesk/HelpScout) for new tickets.
29. Webhook hits a small server (or n8n flow) that calls Claude API with the ticket text + project knowledge.
30. Claude returns a drafted reply.
31. Push the draft to a human review queue. One-click approve and send.

The stack

- Claude API — for any automation beyond chat
- Slack workspace (optional but recommended for team rollout)
- Helpdesk tool — Intercom, Zendesk, HelpScout, Front
- Knowledge source — Drive, Notion, Confluence, wherever your docs live

> **REAL EXAMPLE** Team member asks in Slack: "How do we onboard enterprise SaaS clients?" → Claude pulls the relevant SOP, references the last 2 enterprise onboardings as precedent, and drafts an intro email — all in 30 seconds. Answer cites the SOP doc.

Section 4 — Multi-agent system

> **WHAT YOU SHIP** An AI assembly line. One Claude agent writes, another reviews tone, another formats for the platform, another schedules. Five agents, one human supervisor (you), 5× the output.

What you're building

Take a multi-step workflow you do today — say, content production — and split it across multiple specialized Claude agents. Each agent does one thing well. They handoff outputs sequentially. A human approves at the critical handoffs.

The result: a 2-hour task becomes a 15-minute supervisory review.

Step-by-step

Step 1 — Map your workflow first

Write down your current process. Be specific about time per step. Example for content production:

- Idea → write draft (90 min)
- Edit for tone and clarity (30 min)
- Format for LinkedIn — hook, structure, line breaks (15 min)
- Schedule and post (5 min)
- Track performance afterwards (10 min)

Total: 2.5 hours per piece. Each of those steps becomes one agent.

Step 2 — Set up the stack

32. Sign up at n8n.io — Cloud (paid, no setup) or self-hosted (free, more work). For a first build, use Cloud.
33. Get your Claude API key: console.anthropic.com → API Keys → Create. Set a spending limit so you can sleep.
34. Sign up for Pinecone (pinecone.io) — free tier handles up to 1M vectors. Create an index named content-memory.
35. Airtable free account — create a base called Content Pipeline with tables for Drafts, Approved, Performance.
36. Slack workspace with a bot user for notifications (use Slack Bolt or n8n's Slack node).

Step 3 — Build the flow in n8n

Drag-and-drop these nodes in order:

Node 1 — Webhook trigger

Set this up to receive a content idea — could be a form submission, a Slack message, or a manual run for testing.

Node 2 — Writer agent (Claude API)

Prompt template:

```
You are the writer agent. Write a 500-word first draft about:

{{ $json.topic }}

Use the writing-voice skill from the attached system context.
Return only the draft. No preamble.
```

Node 3 — Tone editor agent (Claude API)

Input: Node 2's output. Prompt:

```
You are the tone editor. Review this draft against our voice rules.
Fix any drift. Keep the structure. Don't rewrite the ideas — just
tighten the language.

Draft:
{{ $node["Writer"].json.draft }}

Return only the edited draft.
```

Node 4 — LinkedIn formatter agent

```
You are the LinkedIn formatter. Take this edited draft and reformat
for LinkedIn:
- Strong 1-line hook
- 3–5 short paragraphs with line breaks between
- Specific CTA in last line
- No hashtags

Draft:
{{ $node["Tone editor"].json.draft }}
```

Node 5 — Save to Airtable

Push the final formatted post to your Content Pipeline → Drafts table. Status: "pending review".

Node 6 — Slack notification

Post to your #content-review channel: "New draft ready: [first 100 chars]... → Approve in Airtable."

Node 7 — Conditional: if approved

Watch the Airtable record. When status flips to "approved", trigger the next node.

Node 8 — Publish (Buffer/Hypefury/Make API)

Post to LinkedIn via your scheduler of choice. Log post ID back to Airtable.

Step 4 — Add shared memory (Pinecone)

For agents to share context across the line:

- Each agent writes its output as an embedding to your content-memory index.
- Each subsequent agent retrieves relevant context (past similar drafts, voice examples) before working.
- This means by the time the formatter agent runs, it knows the original idea AND the writer's draft AND the tone editor's notes.

> **MEMORY MATTERS** Without shared memory, each agent only sees the immediate previous output. With it, the whole chain becomes coherent — and you can layer on agents that reference any prior step.

Step 5 — Test end-to-end

Run one piece of content through the entire flow before going live:

- Does the writer match your voice? If not — refine the writing-voice skill.
- Does the tone editor over-correct? If yes — loosen its instructions.
- Does the LinkedIn formatter break long sentences? If no — add that to its prompt.
- Does Slack ping correctly? Did the Airtable status update propagate?

Step 6 — Add human-in-the-loop checkpoints

At critical decisions (final approval before posting), pause the flow and ping Slack with: "Approve this post? [Y/N]". You approve in 30 seconds via Slack reaction. Flow continues automatically.

This is the difference between a system you trust and one you babysit.

The production stack

- Claude API — the brain. Every agent runs on it.
- n8n — visual orchestration, no-code.
- Pinecone — shared memory vector DB.
- Airtable — source of truth, approvals, dashboards.
- Slack — notifications + human checkpoints.

> **THE OUTCOME** One human supervises four agents. 5× output. Zero burnout. Full control because you sign off at each critical step.

Section 5 — Memory + workflow chains

> **WHAT YOU SHIP** Claude that remembers your business — customer context, brand voice, past decisions, project state. Plus multi-step workflows that pass context cleanly between every action. The compounding moat.

Why this is the final piece

Tools and prompts are commodities — anyone can copy them. Your business memory + chained workflows is not. The longer you operate it, the more leverage compounds. After 6 months, your competitors can't catch up without rebuilding the same context from scratch.

Step-by-step

Step 1 — Turn on Memory in Claude

37. Settings → Memory → toggle ON.
38. Click "Generate memory from chat history" if you have past chats to mine.
39. Claude scans your history and builds an automatic memory file. Review it — edit anything wrong or outdated.

Step 2 — Build a business-context.md doc

Create one master file that captures your business intelligence:

```
---
name: business-context
description: Persistent context about [Company], our customers,
  voice, and active decisions.
---

# Company Context

## Who we serve
[Your ICP — be specific. Industry, role, company size, pain point.]

## Our voice
[Reference your writing-voice.md, plus 3 reasons we DON'T sound
generic AI.]

## Active customer accounts
- [Customer 1]: signed [date], using [plan], champion is [name]
- [Customer 2]: ...

## Recent strategic decisions
- [Date]: Decided to focus on [segment] because [reasoning].
- [Date]: Pricing changed to [model] because [reasoning].
- [Date]: Killed feature [X] because [reasoning].

## Active projects
- [Project name]: status, owner, deadline, blockers.
```

```
## What we don't do
[The 3–5 things we explicitly say no to. This matters as much as
what we do.]
```

Step 3 — Add it to every Project

Every Claude Project you build — add business-context.md to its Knowledge. Now Claude has your business memory in every context, automatically.

Step 4 — Build your first workflow chain

A workflow chain = a multi-step action where each step's output feeds the next. Example: an inbound lead chain.

- Trigger: New email lands from a sender you don't know.
- Step 1 — Claude: Research the sender (uses lead-research.md skill).
- Step 2 — Claude: Draft a personalized reply (uses writing-voice.md + research output).
- Step 3 — Action: Save the lead + context to your CRM (via MCP).
- Step 4 — Notification: Slack ping with the draft for your approval.

Build this in n8n with: Gmail trigger → Claude API (research) → Claude API (reply) → HTTP request to CRM → Slack message.

Step 5 — Set up vector storage for chained memory

If your chains span multiple sessions or weeks, use Pinecone:

40. Create a Pinecone index for your business data (call it business-brain).
41. Embed every customer interaction, every decision log, every chain run.
42. Retrieve relevant context at the start of each chain step.
43. Connect to Claude via MCP or directly in the n8n flow.

Step 6 — Add error handling and human checkpoints

- Slack alert if any chain step fails. Don't let silent failures pile up.
- Auto-route low-confidence outputs to a human review queue.
- Log every chain run in Airtable for debugging and audit.
- Set spending alerts on your Claude API account so a runaway loop doesn't bankrupt you.

Why this compounds

Every interaction Claude has with your business adds to the context. Every decision logged becomes precedent for the next. Every customer touchpoint is one prompt away. Six months in, you have a system that knows your business better than any new hire could — and onboards new humans in a single day.

> **THE MOAT** Tools change. Prompts get copied. Memory + chained workflows are the only Claude work that compounds permanently. Build them last, but build them seriously.

How to ship this — without burning out

The build order (5 weeks)

44. **Week 1:** Build 2–3 Skills. Start with writing-voice, lead-research, repurpose. Use them daily.
45. **Week 2:** Install 5 MCPs. Test workflows. Chain Firecrawl + Brave + Notion for one real intel pipeline.
46. **Week 3:** Build the Internal Assistant. Roll out to your team via Slack.
47. **Week 4:** Build one Multi-Agent flow — pick content production or research. Run it 10 times.
48. **Week 5+:** Layer in Memory + Chains across all of the above. This is where the compounding starts.

Time saved per week (after build)

- Skills: 3–5 hours/week
- MCPs: 5–10 hours/week (depending on your tool stack)
- Internal Assistant: 5–15 hours/week (scales with team size)
- Multi-Agent: 10–20 hours/week (replaces repetitive multi-step work)
- Memory + Chains: compounding leverage — hard to quantify, but eventually 20+ hrs/week of context recovery and decision recall

Pitfalls to avoid

- Building all 5 at once. Ship one fully before starting the next.
- Over-engineering your first Skill. Start with 1,500 words. Iterate based on actual outputs.
- Ignoring Memory. Without it, every other system stays brittle and you re-explain context every chat.
- Skipping the human-in-the-loop. Agents drift. Catch it before it ships.
- Skipping testing. Each agent and chain needs 5+ real test runs before going live.
- Underestimating onboarding. Once Skills and Memory are real, a new hire onboards in a day instead of a month. Make sure your handover docs reflect that — your tribal knowledge is now in Claude.

Where to go from here

Once these five systems are running, the next layer to add:

- Cowork — Anthropic's desktop agent for file-system and task automation. Most users don't know it exists.
- Claude in Chrome — browser agent that takes actions in your tabs. Logs into SaaS, fills forms, exports reports.
- Custom MCPs — build your own connector for internal tools that don't have one yet.
- Fine-tuned voice systems — at scale, train custom Claude models for your highest-leverage workflows.

Ship it.

The five systems above are the difference between using AI and operating with it.

MEGHA SHARMA