AI EDGE HQ · FREE RESOURCE

# CLONE YOUR BRAIN — The Complete Starter Kit

Everything from the video in one file. Six steps, three copy-paste prompts, one template, one worked example. By the end you have a folder of files that AI agents can execute your actual workflows from — with you kept in the loop at the decisions that matter.

---

## THE SYSTEM

| Step | What happens |
|---|---|
| **01 — AUDIT** | AI interviews you. Every recurring workflow: trigger, steps, time, judgment. |
| **02 — TRIAGE** | Clone Score = hours/month × repeatability. Heavy + mechanical goes first. |
| **03 — SPECS** | Each top-5 workflow becomes one file: the flow, the rules, the review gates. |
| **04 — FOLDER** | Specs live in one directory with a README written FOR AGENTS. |
| **05 — MAPS** | Every spec renders as a living diagram. Link color = who runs the step. |
| **06 — RUN IT** | Fresh agent + the folder + one line. It executes. You review at the gates. |

### The two rules

1. **A workflow isn't cloned until an agent with ZERO context can run it from the file alone.** If the agent needs to ask you what "the usual doc" means, the spec failed. Fix the spec, not the agent.
2. **High-judgment steps don't get skipped — they get HUMAN-GATED.** Agents produce, you decide. Every ⋄ REVIEW GATE in a spec is a moment where work stops and waits for you.

### Order of operations (first session, ~90 minutes)

1. Paste the Life Audit prompt (Step 01 below) into Claude. Answer honestly for ~15 minutes. Keep the output.
2. Read your Clone List. Pick workflow #1 — the highest Clone Score.
3. Paste the Spec Writer prompt (Step 03), then describe that workflow out loud (voice mode works great — messy is fine). Answer its gap questions. Keep the spec it writes — you'll file it in a moment.
4. Create your brain folder using the structure in Step 04. Save the spec into it as `specs/01-<workflow-name>.md` (the number = its Clone Score rank).
5. Feed the spec to the Map Generator prompt (Step 05). Save the HTML next to it. Open it. That's your business, drawn.
6. Open a **fresh** chat with no context, attach the folder, and paste the cold-execution line (Step 06). Watch it run. Where it stumbles, the spec has a gap — fix the file.

Repeat steps 3–6 for the rest of your Clone List. One spec per workflow. The folder IS the brain.

**Works with:** Claude (claude.ai), Claude Cowork, Claude Code — or any agent that can read files. The prompts are model-agnostic; the system is the point.

---

# STEP 01 — THE AI LIFE AUDIT

Paste everything between the markers into Claude. Answer its questions like you're talking to a smart operations person. Takes about 15 minutes.

**What Claude will do when you paste this:** it asks you ONE question and stops. You answer, it asks the next — 12 to 15 questions total, digging into whatever sounds heavy. Then it delivers three things in one message: a scored table of every workflow it found, THE CLONE LIST (your top 5, each with what an agent runs vs. where you stay in the loop, and exactly what an agent would need to know), and a paragraph totaling what this costs you per month and year. If your answers are vague, it pushes. If a workflow is chaos, it says so. Don't rush your answers — the whole system downstream is built from this output, so save it somewhere.

▼ COPY FROM HERE

You are my operations auditor. Your job: interview me, map every recurring workflow in my work and life, then tell me exactly which ones to hand to AI agents first.

RULES FOR THE INTERVIEW:

- Ask ONE question at a time. Never send a list of questions.
- Keep questions short and conversational. Follow up on anything vague — if I say "I do outreach," ask what that actually looks like step by step.
- Push past my job title. You're hunting for RECURRING work: things I do daily, weekly, or every time a trigger happens (new client, new video, new order, Monday morning).
- For each workflow you find, get four things before moving on: 1. TRIGGER — what kicks it off 2. STEPS — what actually happens, in order, including which tools I touch 3. TIME — minutes per run, and runs per week or month 4. JUDGMENT — which steps are real decisions vs. mechanical moves anyone could follow
- Cover these areas in order, but follow the thread when something sounds heavy: core work/business, communication (email, DMs, messages), content or marketing if I make any, admin (invoices, scheduling, files, tracking), and personal ops (planning, research, purchases).
- After roughly 12–15 questions, or when you've found 8+ workflows, stop interviewing and deliver the audit.

THE AUDIT (your final output):

1. A table of every workflow found, with columns: Workflow · Trigger · Hours/month · Repeatability (1–5: does it follow the same steps every time?) · Judgment (1–5: how much of it is real human decisions?) · Clone Score. * Clone Score = Hours/month × Repeatability. Highest score = clone first. *

Anything with Judgment 4–5: mark it "HUMAN-GATED" — the agent does the work, I keep the decision.

2. THE CLONE LIST: the top 5 workflows ranked by Clone Score. For each one, write: * One line on what an agent runs vs. where I stay in the loop * One line starting "TO CLONE THIS, an agent needs to know:" listing the specific knowledge, files, examples, or access it would need from me
3. One paragraph titled WHAT THIS IS COSTING ME: total hours per month across the clone list, and what that is per year.

Do not flatter me. If a workflow I describe is chaotic or shouldn't exist at all, say so. Ask your first question now.

▲ COPY TO HERE

---

# STEP 02 — TRIAGE: READING YOUR CLONE SCORE

The audit already did the math. This section is how to read it — and the one mistake to avoid.

### The formula

**Clone Score = Hours/month × Repeatability (1–5)**

Heavy AND mechanical wins. A 20-hour/month workflow that follows the same steps every time (Repeatability 5) scores 100 and gets cloned tonight. A 20-hour/month workflow that's different every time (Repeatability 1) scores 20 and waits.

### The Judgment column is not a blocker

High Judgment (4–5) does NOT mean "can't be cloned." It means **HUMAN-GATED**: the agent does all the mechanical work in the workflow, and the spec inserts a ⋄ REVIEW GATE at every real decision. You approve; it continues. Most valuable workflows are exactly this shape — 80% mechanical moves wrapped around 2–3 real decisions.

If a workflow is ALL judgment (Repeatability 1, Judgment 5 — e.g. "negotiating," "creative direction"), it's not a clone candidate. It's your job. Leave it off the list.

### The mistake to avoid

**Don't clone everything. Clone five things.** Everyone who tries to digitize their whole life at once bounces off the enormity and quits by Thursday. The audit picked your five. Start with number one. When it runs cold-execution clean (Step 06), move to number two.

### The cost paragraph

The audit ends with WHAT THIS IS COSTING ME — total hours across the clone list, monthly and annualized. Write that number somewhere you'll see it. That's the payroll you're currently paying yourself to be your own intern.

---

# STEP 03 — THE SPEC WRITER (VOICE)

Take one workflow off your Clone List. Paste everything between the markers into Claude, then just talk — voice mode is perfect for this. Describe the workflow the way you'd explain it to a new hire: messy, out of order, with tangents. That's fine. The prompt's job is to turn your mess into a file an agent can execute.

**What Claude will do when you paste this:** it waits for your full description first — talk as long as you need. Then it asks you at most 5 short gap questions, one at a time, only about things that would actually block an agent. Then it writes the complete spec in the exact format below: your flow as numbered steps with a named executor on every one, a ⋄ REVIEW GATE wherever you said you check or approve things, and `[CONFIRM: ...]` placeholders for anything you didn't nail down — it will not fill gaps with plausible guesses. If your process is genuinely chaotic, it tells you that before writing anything. Copy the finished spec into a file — that file is the clone.

▼ COPY FROM HERE

I'm going to describe one of my recurring workflows out loud. It will be messy and out of order. Your job: turn it into a spec file that an AI agent with ZERO context about me or my business could execute correctly from the file alone.

RULES:

- Listen to my whole description first. Then ask me a MAXIMUM of 5 gap questions — only about things that would actually block an agent (missing triggers, undefined terms, unnamed tools, unclear owners). One at a time.
- NEVER invent details. Anything I didn't confirm gets an explicit placeholder in the format `[CONFIRM: what's missing]` — never a plausible guess. A wrong number in a spec is worse than a blank.
- Every step must name EXACTLY who or what executes it: me, a named person, a named AI agent, or a named tool. "Someone sends the email" is a failed step.
- Any step where I said I check, approve, decide, or sign off becomes a ⋄ REVIEW GATE — a hard stop where the agent waits for me. Do not soften my decisions into agent steps.
- If my process is chaos — steps that contradict each other, work that shouldn't exist, a loop with no exit — say so FIRST, before writing the spec. Then spec what I actually described, with the problems flagged.
- Length: 40–60 lines. Tight. An agent reads this, not a novelist.

OUTPUT FORMAT (exactly this structure, as a markdown file I can save):

`<WORKFLOW NAME>` — `<one-line descriptor>`

---

**Purpose:** what this workflow produces and why it exists. **Trigger:** what kicks it off. **Cadence:** how often it runs. **Involved:** every executor — people in

CAPS, agents/tools named, with their role in one word each.

### THE FLOW

Numbered steps. Each step: `[EXECUTOR]` does X → produces Y. Insert `⟡ REVIEW GATE — WHO: what they're checking` between stages wherever I check things.

### AGENT BRIEFING

The rules, definitions, and hard constraints an agent needs that aren't steps: what words mean, what quality bar applies, what never happens, which fields are load-bearing downstream.

### INPUTS & ACCESS

Tools, files, folders, and access the workflow touches.

### OUTPUTS

What exists when the workflow is done, where it lands, and what runs downstream from it.

### FAILURE RULES

What the agent does when things go wrong: missing data, no response, a gate not passed. Every failure rule ends in either a safe default or "flag me" — never a guess.

When you have my description and your gap answers, write the complete spec. Name the file `<workflow-name>.md`.

▲ COPY TO HERE

#### The blank spec skeleton (for reference)

```
# <WORKFLOW NAME> — <one-line descriptor>

**Purpose:** <what this produces and why it exists>
**Trigger:** <what kicks it off> **Cadence:** <how often>
**Involved:** <PERSON> (role) · <AGENT/TOOL> (role) · <TOOL> (role)
**MAP:** [../maps/<workflow-name>.html](../maps/<workflow-name>.html)

## THE FLOW

1. `[EXECUTOR]` does X — with which tool, from which input → produces Y
2. `[EXECUTOR]` does X → produces Y
3. ⟡ REVIEW GATE — <WHO>: <what they're checking and what kills the step>
4. `[EXECUTOR]` does X → produces Y
...
n. `[EXECUTOR]` hands off → <downstream spec or final output>

## AGENT BRIEFING

- <A rule that isn't a step: what a term means, a quality bar, a hard filter.>
- <What NEVER happens, and why.>
- <Which field is load-bearing downstream — get it right here or break things later.>
- <Placeholder discipline: unconfirmed values look like [CONFIRM: fee currency].>

## INPUTS & ACCESS

- Tools: <list> · Files: <folders/paths> · Access: <accounts, calendars, inboxes>

## OUTPUTS

<What exists when done, where it lands, what runs downstream.>

## FAILURE RULES

- <Failure condition> → <safe default or "flag me">
- <Failure condition> → <safe default or "flag me">
- Nothing irreversible happens without a passed gate.
```

---

# STEP 04 — THE BRAIN FOLDER

The folder isn't a step — it's the container. A folder of specs IS the brain. Build it once, and every new spec just gets dropped in.

#### The structure

```
my-brain/
├── README.md            ← written FOR AGENTS, not for you (template below)
├── specs/
│   ├── 01-<workflow>.md ← one spec per Clone List workflow, numbered by priority
│   ├── 02-<workflow>.md
│   └── ...
├── maps/
│   ├── 01-<workflow>.html  ← the interactive map for each spec (Step 05)
│   └── ...
└── context/             ← OPTIONAL: reference files specs point to
    ├── voice-examples/  ← e.g. 3 emails that sound like you
    └── templates/       ← e.g. the invoice template a spec references
```

### Rules

1. **One spec = one workflow = one file.** Never merge two workflows into one spec — the agent can't tell where one ends.
2. **Numbering = priority.** `01-` is your highest Clone Score. An agent told to "start" starts at 01.
3. **The README is for agents.** It's the first thing a fresh agent reads. Use the template below.
4. **`context/` earns its place.** Only add reference files a spec explicitly points to. A spec that says "match my voice — see context/voice-examples/" works. A dumping ground of 400 files doesn't.
5. **Specs reference each other by filename.** When one workflow hands off to another ("→ runs `02-client-onboarding.md`"), that's a real, followable link for the agent.

**Where it lives:** anywhere an agent can read files — a local folder for Claude Code / Cowork, a project folder in claude.ai, a repo. Keep it in version control if you can; your brain deserves a commit history.

### Template: README.md (for agents)

Copy this into your brain folder as `README.md` and fill in the `<brackets>`. This file is read by AI agents, not humans — write it like an operating manual.

```
# README — FOR AGENTS

You are operating inside <NAME>'s workflow system. This folder is the complete,
authoritative description of how their recurring work runs. Read this file fully
before executing anything.

## What this folder is

- specs/ contains one file per workflow. Each spec is self-contained: an agent
  with zero outside context must be able to execute it from the file alone. If
  you find you need information a spec doesn't contain, that is a spec defect —
  flag it, do not guess.
- Specs are numbered by priority. Told to "start" with no other instruction?
  Start at the lowest number.
- maps/ contains a visual diagram of each spec. They are for humans; you don't
  need them.
- context/ contains reference files that specs explicitly point to. Only open
  what a spec sends you to.

## How to read a spec

- THE FLOW is the execution order. [EXECUTOR] tags name who runs each step —
  only execute steps tagged for an agent. Steps tagged with a human's name are
  theirs, not yours.
- ✧ REVIEW GATE is a hard stop. Produce everything up to the gate, present it,
  and WAIT. Never continue past a gate on your own judgment, no matter how
  obvious the approval seems.
- AGENT BRIEFING contains rules that override your defaults. If a briefing rule
  conflicts with what seems efficient, the briefing wins.
- FAILURE RULES tell you what to do when reality doesn't match the spec. Every
  failure path ends in a safe default or "flag <NAME>" — never improvisation.

## Standing rules (apply to every workflow)

1. Never invent facts, numbers, dates, or fees. Anything unconfirmed stays as an
   explicit [CONFIRM: ...] placeholder. A visible gap is correct; a plausible
   guess is a failure.
2. Nothing external sends without a gate. No email, message, post, or file
   leaves this system to a third party unless the spec's flow explicitly passed
   a review gate for it.
3. Match the voice. Anything written in <NAME>'s name follows the examples in
   context/voice-examples/ <DELETE IF UNUSED>.
4. <YOUR RULE — e.g. "All money amounts include currency.">
5. <YOUR RULE>

## Escalation

When blocked, confused, or facing a decision the specs don't cover: stop, state
what you have, state what's missing, and ask. One clear question beats ten
assumptions.
```

---

# STEP 05 — THE MAP GENERATOR

Attach one spec file (or paste it below the prompt), then send. One spec per map — don't attach two.

**What Claude will do when you paste this:** first it replies with a plain-language list of every node, connection, review gate, and feedback loop it

extracted from your spec, and flags anything ambiguous — read this, because if the extraction is wrong the map will be wrong, and one correction here is cheaper than a rebuild. Then it writes a single self-contained HTML file. In claude.ai it appears as an artifact you can preview live in the chat — use the download button and save it as `maps/01-<workflow-name>.html` next to your spec, then open it in any browser. Click the flows in the right sidebar to light up each path; hover any node to see its connections. Everything on the map lives in one data object at the top of the file, so to fix a label or add a step you edit the data, not the code — or just tell Claude what to change.

```
▼ COPY FROM HERE
```

Turn the attached workflow spec into a single-file interactive HTML map. Dark infrastructure-diagram aesthetic. No external dependencies except Google Fonts.

LAYOUT:

- Left-to-right columns following the spec's flow stages. Column headers in small uppercase mono.
- Each step = a rounded node (~178×68px): bold label, one-line mono sub-note, and a small badge pill naming EXACTLY who runs it (every agent, model, tool, or person from the spec — e.g. CLAUDE, GPT, CLAY, [MY NAME], TEAM).
- Node accent color by executor type: violet = AI agent · teal = other tools/automation · amber = human · slate = external signal · green = revenue/output. Legend top-right.
- Curved bezier connectors between every step that feeds another. NOTHING may be an orphan — every node needs a producer or a consumer. If the spec implies a link, draw it.
- Every `⟡ REVIEW GATE` in the spec = a dashed amber vertical line between stages with a check-circle labeled REVIEW + who.
- If any output feeds back into an earlier step (analytics, renewals, learnings), draw it as an amber dashed arc sweeping UNDER the whole graph, with an uppercase mono label naming the loop.

INTERACTION:

- Right sidebar: 3–5 named "flows" (end-to-end paths through the map). Clicking one dims everything else, lights that path in each edge's executor color, and animates small glowing particles traveling along the active edges (SVG animateMotion).
- Auto-select the longest flow ~1 second after load.
- Selected flow shows numbered steps in the sidebar — one bold line + one sentence each.
- Hovering any node lights its connections, even while a flow is selected.
- Edge color = the color of the node that PRODUCES the work. Idle edges visible (~40% opacity, dashed, slowly animating); active edges solid with a soft glow.

CONSTRAINTS:

- Must fit one screen: SVG scales with viewBox + preserveAspectRatio, no scrolling.
- Fonts: Space Grotesk (labels) + JetBrains Mono (everything technical). Background near-black (#0A0E14) with a faint dot grid.
- Footer: node count · connection count · gate count · loop count.
- Keep all nodes/edges/flows in ONE data object at the top of the file so I can edit the map by editing data.
- Respect prefers-reduced-motion.

Before writing code, list the nodes, edges, gates, and loops you extracted from the spec and flag anything ambiguous. Then build.

```
▲ COPY TO HERE
```

---

# STEP 06 — COLD EXECUTION: THE PROOF

A workflow isn't cloned until a FRESH agent — new chat, zero context, no memory of you — can run it from the folder alone. This is the test. Run it on every spec before you trust it.

### How to give the agent your folder

- **Claude Code / Claude Cowork:** open a new session in (or pointed at) your `my-brain/` directory. It can read the files directly.
- **claude.ai:** create a Project called "My Brain," upload `README.md` and your specs (plus anything in `context/`) as project files, then start a fresh chat inside that project.
- Either way: new session, no prior conversation. If the agent already knows your business from earlier chats, the test proves nothing.

### The one line

Open a brand-new session. Attach or point the agent at your brain folder. Send exactly this:

```
▼ COPY FROM HERE
```

Read README.md, then execute `specs/01-<workflow-name>.md`. Stop at every review gate and wait for me.

```
▲ COPY TO HERE
```

**What Claude will do:** read the README, read the spec, then start executing the agent-tagged steps in order — and go silent at the first ⟡ REVIEW GATE, presenting its work and waiting for your reply. That silence is success. It should NOT run the whole workflow end to end; a gate-respecting stop is the system working.

If the spec is good, the agent works. If the agent asks a question the folder should have answered — you found a gap. **Fix the file, not the chat.** Answering in-chat patches this run; patching the spec fixes every future run.

### The fuller version (first run of a new spec)

For a spec's maiden run, use this instead — it forces the agent to expose gaps before doing anything:

▼ COPY FROM HERE

Read README.md, then read `specs/01-<workflow-name>.md` fully. Before executing:

1. List every step you'll run vs. every step that belongs to a human.
2. List every `[CONFIRM]` placeholder and anything else you'd need to guess.
3. Name the first review gate you'll stop at.

If your lists match my expectations I'll say GO. Then execute, stopping at every gate.

▲ COPY TO HERE

### Scoring the run

- **Clean run, gates respected, output right** → spec is live. Move to the next Clone List item.
- **Agent asked a fair question** → spec gap. Add the answer to AGENT BRIEFING or INPUTS, re-run cold.
- **Agent guessed something** → failure rule missing. Add the failure rule ("missing X → flag me"), re-run cold.
- **Agent blew through a gate** → the gate isn't written as a hard stop. Rewrite it: `⟡ REVIEW GATE — ME: nothing proceeds until I reply`.

Three clean cold runs = cloned. Cross it off. Next.

---

# WORKED EXAMPLE — A COMPLETE SPEC

This is what a finished spec looks like — a generic weekly newsletter workflow. Yours will name YOUR tools, YOUR people, YOUR rules.

```
# NEWSLETTER ENGINE — Weekly issue, idea to inbox

**Purpose:** Ship one newsletter issue every week that grows the list and
drives one action.
**Trigger:** Monday 9:00. **Cadence:** Weekly; issue sends Thursday 8:00.
**Involved:** ME (decisions, voice pass) · CLAUDE (research, drafting, QA) ·
ESP e.g. Beehiiv/ConvertKit (send, analytics) ·
[CONFIRM: designer or template for header image?]
**MAP:** [../maps/newsletter-engine.html](../maps/newsletter-engine.html)

## THE FLOW

1. [CLAUDE] scans the swipe file + last week's replies + 3 named sources →
   shortlist of 5 topic angles with one-line rationale each
2. ⟡ REVIEW GATE — ME: pick one angle (or kill all five and name my own).
   Nothing drafts before this.
3. [CLAUDE] drafts the issue from the chosen angle: hook → one core idea →
   one example → one CTA, 600–800 words, voice per context/voice-examples/
   → draft v1
4. [ME] voice pass — rewrite anything that doesn't sound like me; tighten to
   one idea → draft v2
5. [CLAUDE] QA pass: subject line ×3 options, preview text, all links
   resolve, CTA present exactly once, no placeholder left behind →
   QA report + final draft
6. ⟡ REVIEW GATE — ME: approve subject line + final draft. This gate is the
   send authorization.
7. [ME] loads approved draft into ESP, schedules Thursday 8:00 →
   scheduled send
8. [ESP] sends Thursday 8:00 → issue delivered
9. [CLAUDE] Friday: pulls opens, clicks, replies, unsubscribes →
   one-paragraph performance note appended to the swipe file →
   feeds step 1 next week

## AGENT BRIEFING

- One idea per issue. Two ideas = cut one. The reader remembers one thing
  or nothing.
- The CTA appears exactly once. More than once reads as selling; zero means
  the issue had no job.
- "Swipe file" = context/swipe-file.md: past subject lines with open rates +
  saved reader replies. It is the voice and topic memory of this workflow.
- Subject lines: under 45 characters, no clickbait words the list has
  already seen (check swipe file).
- Never state a statistic without a source link in the draft.

## INPUTS & ACCESS

- Tools: Claude, ESP account · Files: context/swipe-file.md,
  context/voice-examples/ · Sources: [CONFIRM: the 3 named research sources]

## OUTPUTS

Sent issue Thursday 8:00 + performance note in the swipe file. Downstream:
performance note feeds next Monday's step 1 (the feedback loop on the map).

## FAILURE RULES

- No angle approved by Tuesday 17:00 → issue skips a week; never
  draft-and-send without gate 2. Flag me.
- A link fails QA → hold at step 5, flag me; never send with a dead link.
- Any [CONFIRM] still present at gate 6 → gate fails automatically.
```

*Audit → Triage → Specs → Folder → Maps → Run. One spec per workflow. The folder is the brain.*