

Running a company on AI agents, *with a human on every risky step*

Goutam Soni Case study, October 2026 github.com/goat255/agent-operating-system

I run 4 Step AI alone. I built its data-tools product line with AI coding agents, and for three months agents have run it at full capacity: they read the numbers, research what to build, triage customer issues and draft every change. I approve anything that touches money, customers or production, usually with one tap. This is the system that lets them work unattended without doing damage.

Monthly profit, data-tools product line

USD, 2026. October to the 17th, with month-end pace.



The marketplace's own console figures, after its 20% commission and customers' platform costs, before my own tool subscriptions. Read live 17 Oct 2026. August fell after a price change drove away the largest buyer; pricing is now a gated action (note 1).

The system in seven lines

Counted from the system's own files and tables, 17 Oct 2026.

Scheduled agent jobs	11	note 1
Goal loops with a measured target	7	note 1
Actions agents may not take alone	50	note 1
Approval cards since 11 Oct	185	note 3
Presses, of which applied	85 / 70	note 3
Presses with a written steer	42	note 3
Corrections turned into rules	65	note 5

The problem

An agent that can fix a server can also reboot it at the wrong moment. An agent that can reprice a product can reprice 84 of them. Useful autonomy is narrow: agents measure, plan and do reversible work alone, and everything outward or irreversible goes through a person with enough context to decide in one tap.

The approach

Four parts, close to what a platform team calls a paved road: contracts that say what each job may do (note 1), isolation from secrets and other jobs (note 2), a review gate for every risky step (note 3), and evidence rules so "done" means a check passed (note 4).

1

Contracts: what an agent job may do

Every recurring job that owns an outcome is a **goal loop**. A person writes its contract; only a small runner with no model in it writes its state. The contract holds:

- **A goal with a number and a deadline**, and the command that measures it. A crashing validator is recorded as a failure, never as "no data".
- **Sub-goals with a verification tier**: exit code, number against a threshold, re-measure after a lag, a judge on a *different* model, or a question to me.
- **An action catalogue** with a cost, a cooldown and an idempotency fingerprint, so a retry cannot post a second card or run a second deploy.
- **Forbidden without a gate**: 50 entries across 7 loops. For server health: restart, prune, patch, reboot, delete, stop, firewall, DNS.
- **A budget** (turns, runs per day, wall-clock minutes), **stop rules**, and a **shadow mode**: a new loop plans and records what it would do, and touches nothing until I flip it.

2

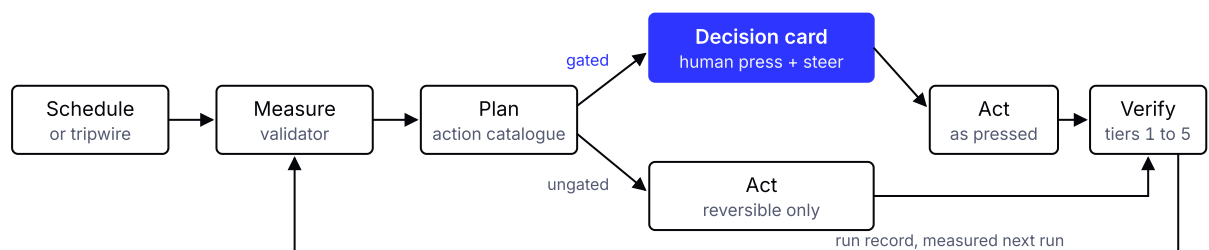
Isolation: what an agent job can see

- **One private repository per cloud job**, built from a manifest, so no job can read another's code, memory or state. The export step redacts and fails closed on anything shaped like a secret.
- **Three environment tiers**: sandboxed with no secrets, an allowlist of my own hosts, or full access, chosen per job.
- **Keys never enter the VM**. The network proxy injects them as headers; server SSH keys never leave my machine.
- **Locally**, secrets are read from the OS keychain by name at the moment of use; tool servers get credentials only through a launcher. A 17-pattern deny list and 14 tool hooks block the dangerous shapes outright.

3

The review gate: how a human stays in the loop

A gated step never runs; it becomes a button naming its exact target ("Reboot server B tonight", never "Proceed"). Presses go through an n8n workflow that verifies the Slack signature, offers an optional free-text steer, records the decision and rewrites the card. The same decisions appear in a PIN-protected web inbox.



Every run ends in an append-only run record; the next run measures against it.

One state rule drives every surface. A decision is pending only if it is unpressed, its card is open, no newer card replaced it, and it is under 7 days old; a late press is answered "already handled" instead of acting on stale context. **Is it used?** Since 11 Oct: 185 cards from 15 jobs, 85 presses, 70 applied, 42 with a written steer. The steers say the cards carry enough context to change a plan, not only to approve it.

4

Evidence: how “done” is decided

The most frequent agent failure is not a wrong action; it is a confident false claim. 14 of the 65 corrections I logged were "done" claims that were not true. Now: loops trust validator output, not narration; a stop-time check blocks a claim of success when the last action changed something and nothing was verified after it; a watchdog checks every scheduled slot for a real artifact, because a scheduler's "succeeded" does not count; configuration rewrites are diffed for dropped keys.

Server health, daily review

10:14 AM · 1 decision

Found Server B has 9% disk left; the image cache holds 41 GB.

Thought Pruning frees about 3 weeks of headroom; nothing else is urgent.

Did Ran 24 checks on both hosts, verified backups are fresh, changed nothing.

[Prune images on server B now](#)[Prune tonight at 2 AM](#)[Not now](#)

Exhibit: every gated step reaches me in this shape, the one I would give a draft pull request. Illustrative card with fictional values; each button names its exact target.

5

What broke, and what each failure produced

Five of the eleven documented incidents; the full log is in the repository.

When	What happened	What exists now
4 to 6 Oct	A usage limit ended 16 scheduled runs early; the scheduler reported success.	Only validator output or a metric proves a run; a code-only watcher checks deliverables.
Aug	A piped install reported exit 0 although it failed.	<code>pipefail</code> everywhere; a completion notice is not evidence.
8 Oct	An agent edited a running script and crashed a long render.	Write a new file, swap atomically.
Aug	Credentials reached a transcript, which cannot be redacted.	Rotate, never redact; secrets read by name and piped.
Recurring	An unattended tool repriced 84 of 88 products.	Pricing gated in every loop that can reach it.

6

Instructions agents read, and research they do

Instructions. A short global file with reference files loaded on demand, 53 project-level files, and 118 task playbooks. A 676-note shared memory injects the four most relevant notes into every prompt, so a correction made once is applied next time. Three agent runtimes share one source of instructions, generated and never hand-edited.

Research. The same agents ran a read-only study of 27 AI-automation communities I belong to, including the largest where people learn n8n: about 21,000 posts, one evidence-based dossier each. Findings: communities capture demand that the founder's content creates; a pinned "start here" thread was the most-engaged post in most of them; members write the large majority of posts; and self-hosted, own-your-stack automation has real appetite with few consistent teachers.

7 Limits of this evidence

- **One human.** Adoption is me plus agents, not a team of engineers. Winning the trust of reviewers who did not build the system is the harder half of the job I am applying for, and I have not done it yet.
- **Not a monorepo workflow.** My agents ship operational changes, reports and small code changes across many small repositories, not issue-to-pull-request at the scale of a large TypeScript monorepo.
- **Measurement is young.** Loops record gap, cost and failures per run, but there is no benchmark of historical tasks yet, and the history is months, not years.
- **Some guardrails are conventions.** Several rules live in instructions the agent follows; the deny list and hooks enforce a subset.

8 How this maps to agentic engineering at n8n

The role asks for	What I would start from
Safe task categories	A forbidden-without-gate list per job, gated actions as buttons naming exact targets, shadow mode before a job may act.
Isolated execution	Per-job repositories, proxy-injected credentials, tiered environments.
Issues to draft PRs	The loop contract: the issue is the goal, CI is the tier-1 validator, the PR is the action, review is the gate.
PR evidence templates	The decision card: found, thought, did, exact targets, what was verified, what still needs a human.
Historical benchmarks	The step I have not built: replay closed issues against agents, score with tier 1 to 4 checks.
Usefulness metrics	Press, steer and applied rates already exist for cards; the same shape works for draft PRs.

9 The first 90 days, if I join

Days 1 to 30: read first	Days 31 to 60: shadow pilot	Days 61 to 90: measure, widen
Map where agents already help in the monorepo and where reviewers push back. Draft repository instructions and three safe task categories with the teams that own them. Nothing autonomous yet.	Selected issues become draft PRs with an evidence template, run in an isolated environment without secrets. Build a benchmark from about 30 closed issues to score the agent before anyone reviews its output.	Track acceptance, rework and reviewer time per draft PR. Widen to a second team only where the numbers and the reviewers both say yes, and write down what the agent must never touch.

The agents were never the hard part. Deciding what they may do alone, and proving what they did, was.

Goutam Soni

Jaipur, relocating to Berlin

[linkedin.com/in/goutamsoni](https://www.linkedin.com/in/goutamsoni)

github.com/goat255/agent-operating-system