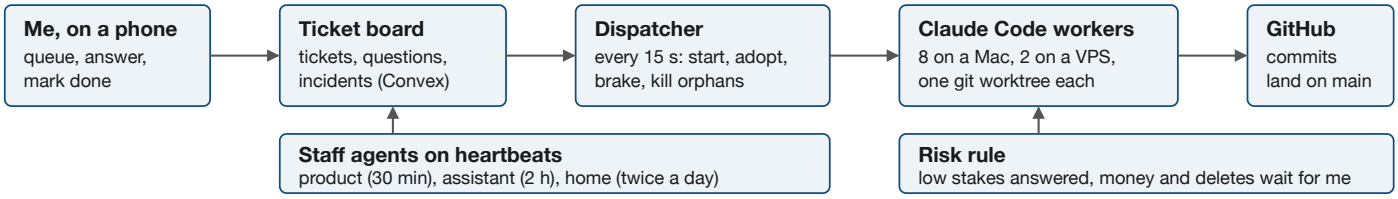
Running agents nobody is watching

A one-page case study by David Chicaiza. Numbers as of 4 Nov 2026. The repo is private, so this page is the proof, and I'm happy to walk through any of it on a screen share.

THE PROBLEM

Most agent demos end when you close the laptop. I wanted agents doing real work on my own systems overnight and while I'm away, stopping only when a decision is actually mine. The model is rarely what fails. What fails is an agent that stops without saying so, a machine that saturates, or a watchdog that lies.

WHAT RUNS



Underneath: git auto-sync every 10 min · secret scan and content gate on every sync commit · dead-man switch · push alerts off the box

A ticket gets a worker: a Claude Code session in its own git worktree that reads the code, ships to main and reports back. Every turn has to end with a question, a done report or a wait, so silence means something went wrong.

THE NUMBERS

6 months
always-on agents, May to Nov 2026, then replaced by what's above

89 days
VPS sync loop, and its longest silence was 10.8 hours

10 workers
running at once at the time of writing

277 tickets
shipped in the pool's first 4 days (it went live 1 Nov), 209 of them on the fleet's own tooling

116 decisions
taken by the risk rule instead of me: 61 questions answered, 55 tickets closed

108 incidents
logged in the first 36 hours of tracking; 71 of 72 lost workers came back on their own

WHAT BROKE, AND WHAT I CHANGED

What happened	The fix
Sep 2026. An expired login made a supervised process exit instantly, and cron stacked 10 supervisors on a 2 vCPU VPS. It ran out of memory and wedged. The alerts lived on the same box, so none got out.	A lock and a backoff on fast exits (15 s up to 600 s). Alerts moved to an outside push service, and the Mac now checks the VPS and pushes when it's unreachable.
1 Nov 2026. Day one of the pool: workers started faster than the Mac could run them. Load hit 828 on 10 cores and some workers never even drew a screen.	A machine brake holds new starts over load 20 and lets go under 15, with at most 2 starts a tick. It has engaged 18 times since, 40 minutes in total, and it held this page's own ticket at load 24.
Jul 2026. The dead-man switch sent 28 identical alerts in 12 hours about a healthy machine: it measured commit age, and a clean tree commits nothing. Once, a hostname changing case read as dead for 53 days.	Every writer leaves a heartbeat file at least every 6 hours. Alerts fire on transitions, announce recovery and tell a dead writer from one waiting on a merge.
Nov 2026. Workers stopped without a word, sat in dialogs or never started, and the board kept showing them as running. Flagged so far: 60 silent stops (some were false alarms), 13 dialogs, 7 no-shows.	Health checks name the stage on each card and restart or nudge once per episode. Every incident gets a row and an ending, including "false alarm".
Jul 2026. A forgotten cron job was a fourth writer to the repo. One swallowed `git stash pop` conflict put corrupted state on main.	Removed it, and added a gate that rejects conflict markers and invalid YAML before anything commits.

WHAT IT TAUGHT ME

- **A long job is a queue, not a long process.** The median worker session is 12 minutes. State lives on the board and in git, so a restart costs little.
- **Alive isn't working.** In August three agents failed every turn when their pinned model ran out of credits, and the health endpoint stayed green. Checks now look at output: session writes, commits, activity lines.
- **A watchdog is one more thing that can be wrong.** Keep it off the box it watches, alert on transitions, and count its false alarms as incidents.
- **Cut what doesn't earn its place.** In October I audited nine chat agents, kept one, removed 969 files and 23 cron jobs, then brought back only what I actually missed.

Limits, so you don't have to ask: no single agent has run for a week straight. I get long work from chains of short workers. The pool is days old. I still queue most tickets myself (320 so far), and most of what it has shipped is its own tooling.

David Chicaiza · dscm0725@gmail.com · linkedin.com/in/david-chicaiza-12a737a1