TradeMagnet

N8N VS. CLAUDE CODE

Everyone says n8n is dead. They're wrong.

The take going around: “n8n is dead, just use Claude Code.” It confuses a **builder** for a **runtime**. Claude Code is an agentic AI harness that writes code, brilliant at *building*, but it hosts nothing and runs nothing once you close the session. n8n is the platform your automations actually *live* on: hosted, connected, triggered, 24/7. Here are five reasons it's still king, and how to point Claude Code at it instead of pretending it's a replacement.

READ TIME	SKILL LEVEL	COMPANION TO
~15 min	Plain English, real mechanics	YouTube tutorial

CHAPTER 01

n8n vs. Claude Code: what each one actually is

The whole “n8n is dead” argument falls apart the moment you define the two things being compared. They're not rivals in the same category, they're different layers of the stack.

n8n: a workflow automation & orchestration platform

n8n is where your automations *live and run*. It hosts them, holds the connections to your other tools, listens for events, and executes the steps, on a server, around the clock, the same way every time. It's the orchestration layer: the thing that makes a bunch of API calls behave like one reliable, repeatable process.

Claude Code: an agentic AI harness

Claude Code is an LLM wrapped in tools, an AI agent that can read, write, and run code. It's an extraordinary *builder*: describe what you want and it'll write the script, wire the logic, debug it with you. But that's where it stops. It doesn't host your automation. It doesn't sit on a server listening for webhooks. It doesn't manage your OAuth tokens or run your workflow at 6am while you sleep. It's **invoked**, it builds, and it leaves.

> **The category error:** comparing n8n to Claude Code is comparing a building to the architect. The architect designs brilliantly, but you don't live in a blueprint. You need the thing that's actually standing up, with the lights on, 24/7.

Side by side

	CLAUDE CODE	N8N
What it is	Agentic AI coding harness	Workflow automation / orchestration platform
Its strength	Building, writes code & logic	Running, hosts & executes, 24/7
Where it lives	Your session / terminal	A server (cloud or self-hosted)
Persistence	Ephemeral, gone when you close it	Always-on, listening for events
Connections / auth	You own all the plumbing	400+ integrations, credentials managed
How it acts	Invoked, you ask	Triggered, the world fires it

So this isn't “which one wins.” It's: **Claude Code builds; n8n runs.** The next five chapters are the concrete reasons that “runs” layer isn't going anywhere.

The five reasons, up front

1. **Hosting**: the always-on runtime your automations live on
2. **Third-party connections**: integrations + credential/OAuth management, baked in
3. **Error handling**: execution logs + dedicated error-handling nodes
4. **Self-hosting**: open source, your infra, scalable, sandboxed
5. **Custom nodes**: build your own, ship a UI to clients

CHAPTER 02 · REASON 1

Hosting: your automations have to live somewhere

Say Claude Code builds you a beautiful automation. Now what? Where does it *run*? You're not going to leave it executing on your laptop with a port forward open to the internet, praying the machine never sleeps. That's not an automation. That's a liability.

An automation is an entire package: code, triggers, schedule, state. All of it has to live on a computer that's always on: a server in the cloud, or one you host yourself. **That's what n8n is.** It's the hosting environment for your automations. Build with whatever you like; n8n is where the thing actually lives and runs.

What “hosted” actually buys you

Hosting isn't just “a box that's switched on.” It's everything that makes an automation trustworthy in production:

- **Always-on triggers.** Webhooks land, schedules fire, app events arrive, 24/7, whether or not anyone's watching. An agent in a session has no inbox, no cron, no open port.
- **A real address.** A stable, public endpoint for your webhooks, not a tunnel to your laptop that dies the moment you shut the lid.
- **One place for everything.** Credentials, run history, and state all live in the one hosted instance, not scattered across whatever machine happened to run the job.

And because every run happens on that always-on server, every run gets captured, which is exactly what makes the next reason possible.

24/7

Claude Code is invoked; n8n is triggered. You *ask* an agent. The world *fires* a hosted workflow, while you’re asleep, on a job, or on holiday.

> **The one-liner:** Claude Code builds the automation; n8n is the address it lives at. Code with no home isn't infrastructure. It's a demo.

CHAPTER 03 · REASON 2

Third-party connections: the part everyone underestimates

n8n ships with an enormous catalogue of ready-made connections, 400+ apps, from ServiceM8 and Xero to Google, Stripe, Slack, and SMS gateways. Tap one, authenticate once, and it's wired in. That convenience hides how much hard work it's doing for you.

If you've ever tried to build your own connector to a third-party API, you already know: it is **no mean feat.** Especially anything using OAuth 2.0: you're suddenly responsible for the token dance, storing refresh tokens, refreshing access tokens before they expire, handling revocation, and retrying the calls that fail because of it. It's fiddly, it's easy to get subtly wrong, and it's the kind of bug that only shows up at 2am three weeks later when a token quietly expires.

> **This is what n8n does best.** OAuth 2.0 flows, credential storage, automatic token refresh, all baked in. You authenticate once; n8n manages the lifecycle. You never write token-refresh code again.

Why this matters against “just use Claude Code”

Claude Code can absolutely *write* the code to call an API. But the moment it does, **you** own all of it, the secret storage, the OAuth refresh, the retry logic, the place those credentials live securely so they're not sitting in a prompt or a plaintext file. n8n gives you that as a managed, reusable layer: store a credential once, use it across every workflow, rotate it in one place.

- **Credentials, vaulted & reused.** Not pasted into prompts or hard-coded, stored once, referenced everywhere.
- **OAuth 2.0, handled.** The token lifecycle is n8n's problem, not yours.
- **Hundreds of integrations.** The connector you need almost certainly already exists, and if it doesn't, see Reason 3 and Reason 5.

CHAPTER 04 · REASON 3

Error handling & execution logs: receipts and a safety net

Real automations fail. APIs time out, tokens expire, a supplier's server has a bad morning. The question isn't *whether* something breaks. It's whether you find out, whether it recovers, and whether you can prove what happened. n8n gives you all three out of the box.

Execution logs: every run, on the record

Every execution n8n runs is recorded. Open execution #3,812 from three weeks ago and see the exact data that flowed through each node: what ran, what failed, what the payload looked like at every step. You can re-run from a point, replay with pinned data, and answer the question every operator eventually asks: *"did the customer actually get that SMS?"* With proof, not a guess.

Error handling nodes: resilience built in

- **Retry on fail.** A flaky API call retries automatically, with backoff and no extra code.
- **Continue on fail.** One bad record doesn't kill the whole batch; the workflow keeps going and you deal with the stragglers separately.
- **Error Trigger workflows.** A dedicated error workflow catches *any* failed execution and does something about it: text you, log it, or park the payload for a retry.

404 **Failure is a feature you design for, not a surprise. n8n assumes things will break and gives you the retries, branches, and alerts to handle it. That's the unglamorous 80% of real automation.**

Why this beats "just use Claude Code": a one-shot script that fails just... fails. No inspectable history, no automatic retry, no alert, unless you build all of it yourself. n8n hands you observability and resilience as table stakes.

The one-liner: n8n doesn't just run your automation; it remembers every run and catches its own failures. That's the difference between a script and a system.

CHAPTER 05 · REASON 4

Self-hosting: why n8n always beat Make

This is the one that separates n8n from closed tools like Make.com: you can **self-host it.** It's open source. You can run it on your own infrastructure, fork it, even rebuild it from scratch if you really wanted to. For a hobbyist that's a nice-to-have. For an AI agency that's serious about this, it's the whole game: you own the platform, the data, and the ceiling.

What self-hosting unlocks

- **Open source.** No black box. Inspect it, extend it, fork it. You're not renting capability you can lose.
- **Environment variables galore.** Out of the box you get a huge surface of config to tune behaviour, security, and limits to your setup.
- **Sandboxing.** Run code in controlled, sandboxed environments. This matters once you're executing custom logic for clients.
- **Real scale (queue mode).** Run with workers backed by Redis and Postgres so many workflows execute concurrently instead of fighting over one process. This is how you go from “a few automations” to “a platform.”

> **The honest caveat:** some enterprise features are gated behind a paid licence (SSO, certain RBAC/scaling niceties). But the open-source core does an enormous amount. Be straight about the line so the argument holds against a sceptic.

The setup the author runs is a self-hosted instance in **queue mode**: Redis + Postgres + worker processes, so multiple workflows run at once, reliably, on infrastructure that's fully under his control. That's not a toy. That's production.

> **Self-hosting is the door.** And behind it is the single most underrated capability in the whole platform, which is Reason 5.

CHAPTER 06 · REASON 5

Custom nodes: the pièce de résistance

Once you self-host, the ceiling lifts entirely: you can build your own **custom nodes.** Not just inline code in a Code node, but a proper, reusable node that shows up in the palette like any built-in one, with its own inputs, its own settings panel, its own polish.

And here's the part worth sitting with. This is *exactly* the kind of thing the "Claude Code replaces everything" crowd is teaching you to build: bespoke agents and tools, wired together with custom frameworks and Python scripts. n8n gives you a framework to do the same thing, except the result is a **node:** you can run any execution you want inside it, and wrap it in a clean UI you can hand to a client.

Anything you can script, you can ship as a node. Reusable, configurable, slotted into the orchestration, with a UI a non-technical client can actually use. That's the difference between a clever script and a product.

Why this kills the "n8n is limiting" jab

- **No capability gap.** If the logic can be written, it can be a node. The platform stops being a constraint.
- **Reusable, not one-off.** Build the node once; drop it into every client's workflows.
- **Client-ready UI.** Your custom logic gets a proper settings panel, not a wall of code the client has to fear.

> **The reframe:** the custom-agent skills the influencers are selling aren't a reason to ditch n8n; they're a reason to build *better n8n nodes*. Which is exactly where the next chapter goes.

CHAPTER 07

Stop demonising it: leverage Claude Code with n8n

Here's the turn. None of this is anti-AI. The smart move isn't to pick a side; it's to use the best *builder* ever made to build for the best *runtime* ever made. Claude Code and n8n aren't rivals; they're a supply chain. Point the agent at the platform and you go faster than either could alone.

Three ways to put them together

1. **Build custom nodes with Claude Code.** The custom node from Reason 5 is just code, which is exactly what Claude Code is great at writing. Have it scaffold the node, wire the inputs, debug the execution. The agent builds; n8n hosts and runs the result.
2. **Build whole workflows with a skill.** Give Claude Code a skill that knows the n8n workflow JSON format, describe the automation in plain English, and it produces a workflow you import, copy, paste, done.
3. **Wire it up over MCP.** Expose n8n to Claude as an MCP server and it can build and manage workflows directly. We do exactly this in the **ServiceM8 Forms guide** ↗: Claude builds a ServiceM8 form end to end through a custom n8n MCP. Same pattern, applied to your own automations.

> **The synthesis:** Claude Code *designs and writes*; n8n *hosts and runs*. Use the agent to author for the platform: custom nodes, full workflows, or over MCP. That's the workflow the “n8n is dead” crowd never shows you.

CHAPTER 08

The verdict: long live n8n

So, is n8n dead? No. The people saying it are comparing a builder to a runtime and calling the runtime obsolete because the builder got smarter. Claude Code got dramatically better at the part n8n was never trying to be: *writing* the automation. It got no better at the part n8n exists to do: **hosting it, connecting it, and running it**, reliably, on infrastructure you control.

The five reasons, in one breath

1. **Hosting**: your automations need an always-on home; Claude Code isn't one.
2. **Connections**: 400+ integrations with OAuth and credentials managed for you.
3. **Error handling**: execution logs, retries, and error-trigger workflows.
4. **Self-hosting**: open source, your infra, scalable, sandboxed.
5. **Custom nodes**: build your own and ship a UI to clients.

And the headline flips: AI doesn't kill n8n; it makes it *more* valuable. You've got the best workflow-author ever built sitting on top of the most flexible runtime, with a brain you can drop into any node that needs one. That's the crown on the thumbnail. n8n is still king; it just hired the sharpest advisor it's ever had.

Claude Code builds. n8n runs. Anyone telling you to throw out the runtime has never had to keep one alive in production.

> **If you remember one line:** Claude Code is invoked and builds; n8n is triggered and runs. Different jobs. You want both.

WHAT NEXT

Want n8n + AI wired up for your business?

Trade Magnet builds and self-hosts n8n automations, custom nodes, third-party integrations, and AI dropped in exactly where it earns its keep, with the hosting, credentials, and error handling done properly. If you've watched the video and want this running for your account, get in touch.

trademagnet.com.au →

New to n8n? Start free at n8n.io ↗, self-host it or use n8n Cloud. Then watch the ServiceM8 Forms guide to see Claude build an automation through a custom n8n MCP.

TradeMagnet

AI automation and apps for industry.

https://trademagnet.com.au · hello@trademagnet.com.au