

A PLAIN-ENGLISH WORKING GUIDE

# /fable-it

## Make Opus behave like **Fable**

---

I loved Fable. The bill didn't love me back. So while I still had access, I reverse-engineered what actually made it better than Opus on long, hands-off jobs and packed those behaviors into one skill you drop into Claude Code. Same discipline. Opus prices.

BUILT BY  
**her0**

GET IT  
**github · see p.08**

FOR  
**Claude Code builders**

# One idea does **the work**

Read this page and you understand the whole skill. The rest is detail.

Fable's edge on long jobs is mostly **discipline**, not magic. And discipline is the part you can copy onto a cheaper model.

When you watch Fable run a long, multi-step task, the thing that stands out is not raw cleverness. It is that it **holds the thread**: it doesn't contradict a decision it made an hour ago, it tests its own work before claiming it's done, it reports honestly when it couldn't verify something, and it resists building things you never asked for. That is behavior, not IQ.

Behavior transfers. You can instruct Opus to hold the thread, to verify, to report honestly, to stop gold-plating. What does *not* transfer is the underlying capability that lets Fable one-shot a genuinely hard problem. That comes from training, not a prompt. So this skill makes Opus **behave** like Fable on long work. It does not turn Opus into Fable, and anyone who tells you a skill can do that is selling something.

## WHAT THE SKILL ACTUALLY GIVES YOU

- ▶ One command. You hand it a goal and a numbered Definition of Done, and walk away.
- ▶ The autonomous posture that keeps Opus moving instead of stopping to ask permission.
- ▶ Three coherence guardrails that stop long runs from quietly drifting apart.
- ▶ An honest report waiting for you, that never fakes a green result it didn't verify.

# 01 The honest claim

What ports across, and what doesn't

Most skills oversell. This one is built on the opposite bet: tell you exactly where the line is, and the part that's left is still worth a lot. Here is the line.

## PORTS TO OPUS ✓

- ✓ Coherence across a long run, holding early constraints
- ✓ Self-verification before declaring a step done
- ✓ Honest, evidence-backed progress reporting
- ✓ Autonomous turn discipline, no needless pausing
- ✓ Restraint, doing the job and not inventing scope
- ✓ One-command orchestration of your existing tools

## STAYS WITH FABLE ✕

- ✕ Raw reasoning ceiling on genuinely hard problems
- ✕ One-shotting a complex system from a thin prompt
- ✕ The deepest long-context retention quality
- ✕ Anything that comes from the weights, not the prompt

## WHY FRAME IT THIS WAY

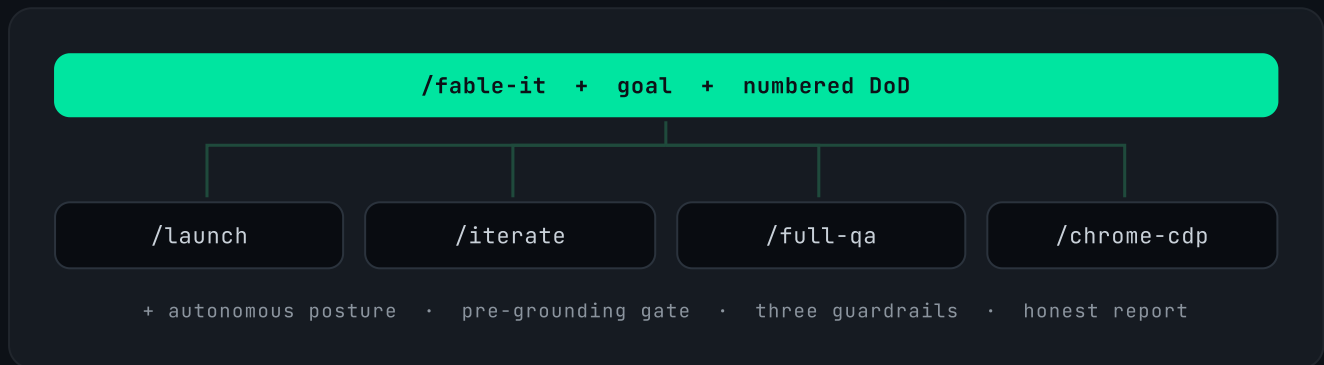
Because "behave like" survives contact with your codebase and "becomes Fable" does not. The first time a reader runs it expecting Fable-grade capability on a hard problem and doesn't get it, an overclaim costs you credibility. The honest claim keeps it. And the honest claim is still: your Opus runs long jobs far more like Fable than it did yesterday.

## 02 One command, four tools

It conducts your stack, it doesn't replace it

If you already run Claude Code with skills like `/launch`, `/iterate`, `/full-qa` and `/chrome-cdp-control`, you know the friction: you re-specify the same posture, the same rules, and the same tool chain in every prompt.

`/fable-it` is the **conductor** over those. You invoke one command with a goal and a DoD; it routes the work to the right tool at the right moment and enforces the behaviors Fable has natively.



### THE DESIGN RULE UNDER THE HOOD

It never pastes a worse copy of your tools' logic. It calls them by name and lets each own its job, the same way you'd reference a teammate's work instead of rewriting it. When you improve `/iterate`, `/fable-it` inherits the improvement. No duplicated source of truth, which is the same discipline it enforces in your code.

## 03 The posture it injects

The behaviors that keep a long run on track

### ● ● ● AUTONOMOUS POSTURE

You are operating autonomously. The user can't answer mid-task, so "Want me to...?" just blocks the work. For **reversible** actions that follow from the goal, proceed. Before ending a turn, if your last line is a plan, a question, or a promise, that work isn't done – do it now. End only when the DoD is met or you're blocked.

That single block replaces the "I'll go to bed, work autonomously, green light" paragraph you write every night. But autonomy has two failure modes, so the skill clamps both with counter-rules Fable's enthusiasm needs too:

### DON'T FAKE CONFIDENCE

Autonomy is not false certainty. Keep flagging what wasn't tested. A confident unverified "it works" is worse than an honest "built, not verified, here's why."

### DON'T GOLD-PLATE

Do the simplest thing the spec and DoD require. "Max completeness" means finish every DoD item, not invent features, abstractions, or refactors nobody asked for.

### ● ● ● PRE-GROUNDING GATE · BEFORE ANY CODE

Read the **real** source of truth first – the actual schema, the real table, the production shape – not your memory of it. State how the data is modeled and where it's stored. For each DoD item, name what you'll verify it against. Then build.

## 04 The three guardrails

Why long, parallel runs stop drifting apart

This is the part that makes it engineering and not vibes. Long autonomous runs fail in three predictable ways. Each guardrail closes one.

### 01 Shared decision contract

When agents run in parallel, every cross-cutting decision, schema shapes, field names, interfaces, lives in one shared file that each agent reads before deciding and writes after. Stops the classic split where a renderer is built for schema A while the connector saves schema B.

### 02 Cross-session interface file

When this run assumes work another session is still building, it doesn't build against an assumption held in its head. It writes the interface, the data shape, the fields, the contract, that both sides agree on. A managed assumption instead of a roulette spin that only fails at integration, hours later.

### 03 Honest per-criterion status

An agent told to "iterate until it works" and leave a success report has every incentive to declare victory. So the report is never binary. Every DoD item gets a state, with evidence.

VERIFIED

IMPLEMENTED • NOT VERIFIED

BLOCKED

#### THE POINT OF GUARDRAIL THREE

It protects you while you sleep. No green result is ever reported off a mock or an assumption. If it couldn't reach the real thing, it says so, plainly, with the reason. That's the Fable behavior worth copying most.

## 05 Install & run

Two minutes, then one command

● ● ● INSTALL

```
# drop the skill folder into your Claude Code skills dir
~/.claude/skills/fable-it/SKILL.md
```

```
# then invoke it with a goal and a numbered DoD
```

```
/fable-it Build the Shopify multi-store connector.
```

```
DoD:
```

1. Backfill shows all records in connector-logs
2. Pooling enabled in UI, records appear in logs
3. New interactions visible on the timeline page
4. ShopifyRenderer shows details for a clicked item

Only two things are required: the **goal** and a **numbered DoD**. Everything else has a sensible default so you stop repeating yourself:

<code>credentials</code>	reads <code>.full.credentials</code> , then <code>.env</code> ; creates tokens via your logged-in browser when needed
<code>tooling</code>	inferred from the DoD — UI criteria route to <code>/full-qa</code> , data criteria to <code>/iterate</code>
<code>parallelism</code>	decided for you; decision-coupled work is kept coherent, not fragmented
<code>report</code>	written to the workspace root, plus a separate credentials file for anything created

### BUDGET FOR LONGER TURNS

It is built to run unattended. Hard jobs take minutes, hands-off jobs can run for a long stretch. A long silence means it's working, not stuck. Point it at a real task before bed and read the report in the morning.

// PIN THIS UP

# The whole thing, **one card**

## THE GOLDEN RULE

- > Brief it, don't boss it. Goal + numbered DoD, then walk away.
- > It conducts /launch, /iterate, /full-qa and /chrome-cdp for you.

## WHAT TO EXPECT

- > Long runs that hold their thread
- > Honest status, never a faked green
- > No scope it invented on its own

## WHAT NOT TO EXPECT

- > Fable's raw capability — that's the weights
- > Magic on a thin prompt — feed it a real DoD
- > A literal "10x". It's discipline, not a miracle

**Get the skill, run it tonight.**

[github.com/your-handle/fable-it](https://github.com/your-handle/fable-it)

Free and open. Drop it in, point it at a real job, and tell us what your Opus did by morning.

*Everyone can be a herO too.*