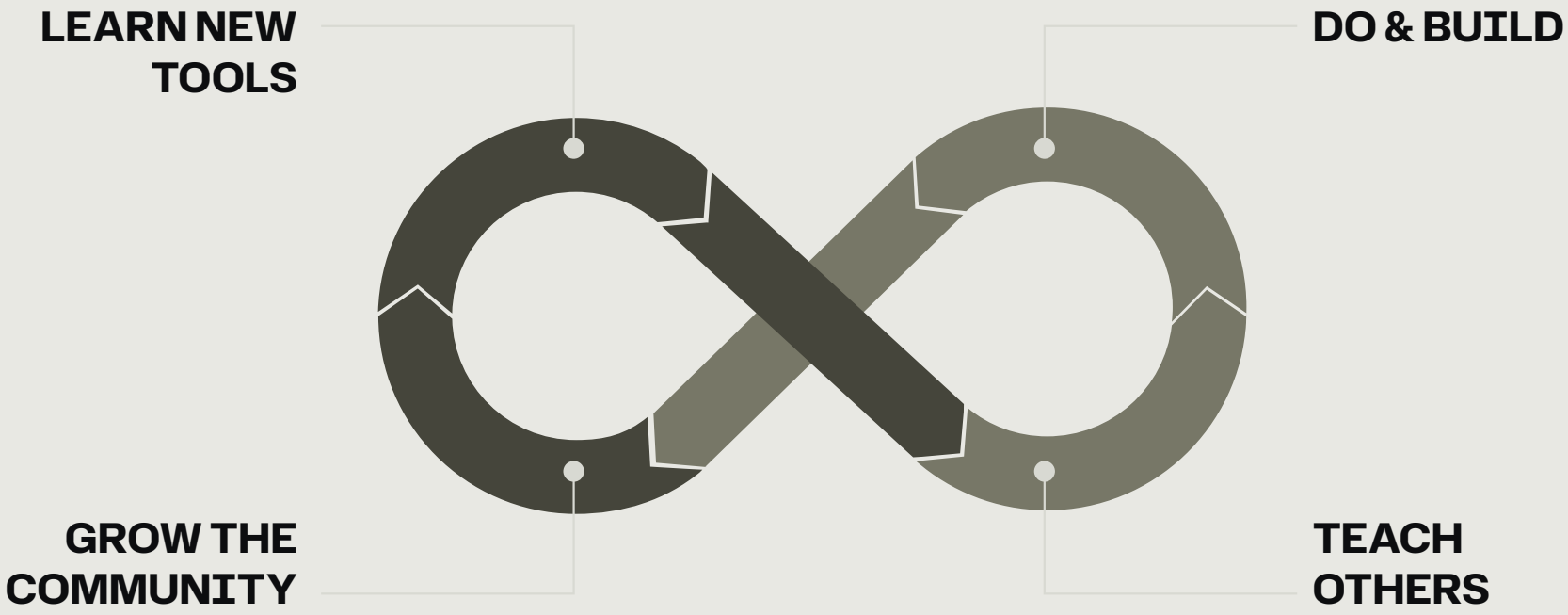
BELL AI FELLOWSHIP

A community built on the **Learn → Do → Teach** framework.

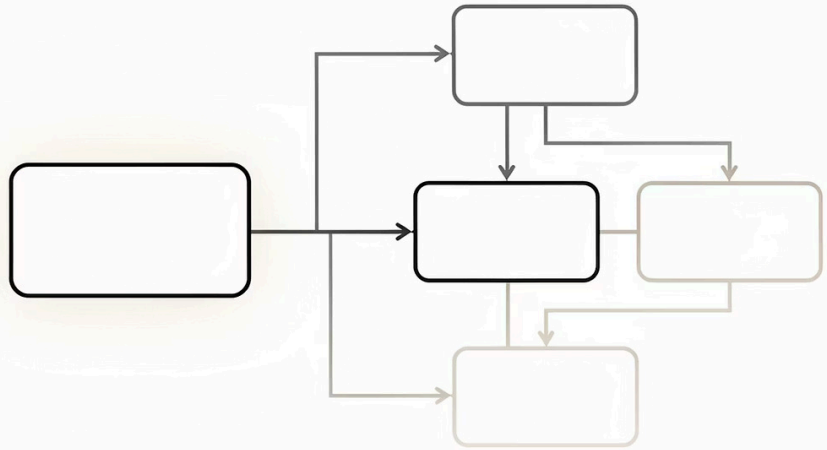


The loop closes when you teach. That's how the fellowship grows.

WHAT IS ORCHESTRATION?

Not just agents talking to agents. Orchestration is how you coordinate and sequence AI work across multiple steps, tools, or agents — turning isolated calls into systems that actually get things done.

> A single prompt is a tool. Orchestration is a system.



SPECTRUM

THE SPECTRUM

Simple to complex — most production value lives in the middle.

1 **PROMPT CHAINING**

Manually pass output between steps

2 **PIPELINE**

Code routes data through stages automatically

3 **TOOL USE**

One LLM decides which tools to call

4 **AGENT LOOP**

LLM runs autonomously with tools until done

5 **MULTI-AGENT**

Multiple agents coordinating in parallel

⚠️ Most production value lives in the middle — not the multi-agent end. Don't over-engineer.

THREE BUILDING BLOCKS

Every orchestration pattern — from a simple pipeline to a multi-agent system — is built from exactly these three things.

1. STATE

Where is everything right now? A database, JSON file, or spreadsheet. The source of truth that everything else reads from.

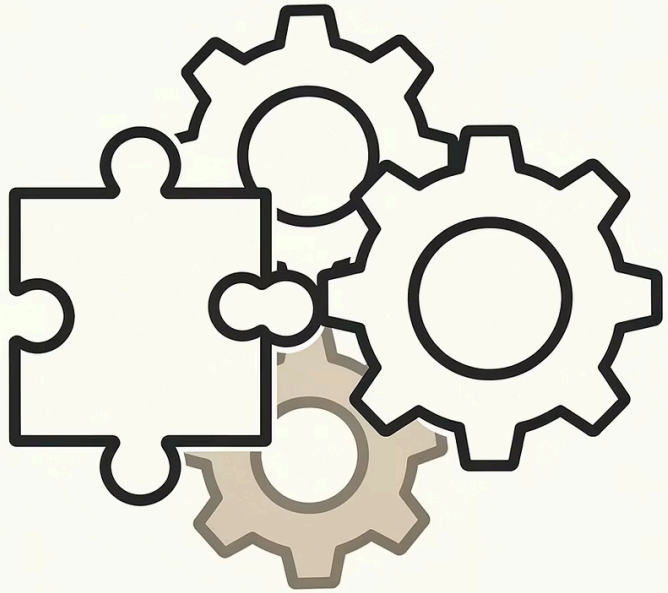
2. ROUTER

Given the current state, what should happen next? Can be hard-coded rules, an LLM classifier, or both.

3. ACTORS

The things that actually do work. API calls, LLM prompts, tool use, or humans in the loop.

Every pattern on the spectrum is just these three pieces composed differently.



AUTONOMOUS CLAUDE CODE

6 LEVELS OF AUTONOMOUS CLAUDE CODE

LEVEL 1: KILL PERMISSION PROMPTS

--dangerously-skip-permissions. Remove the friction. Let it run.

LEVEL 2: CONTEXT WINDOW MANAGEMENT

/clear and /compact at 60%. Keep it focused over long sessions.

LEVEL 3: SUBAGENTS

Separate context windows. 2+ hours of autonomous work per agent.

LEVEL 4: RALPH WIGGUM-LOUP

Stop hook re-feeds the prompt. 27+ hours, 84 tasks completed autonomously.

LEVEL 5: KARPATHY'S AUTORESEARCH

Structured eval loops. 100+ experiments overnight. 25K GitHub stars in 5 days.

LEVEL 6: VPS + OPENCLAW

24/7 operation. Always-on autonomous agent infrastructure.

> Remove the friction. Let it run.

The unlock at every level: give Claude a way to verify its own work.

BELL AI FELLOWSHIP · 90 MIN WORKSHOP

TODAY'S PLAN

01

DEMOS — LEVEL 4 & 5

15 min each. Ralph Wiggum-Loop and Karpathy's AutoResearch. Real systems, live walkthroughs from co-presenters.

02

LIVE BUILD

25 min. We build a working member pipeline together. State machine, router, LLM actor. You leave with something deployable.

03

HANDS-ON EXERCISE

20 min. You level up. Beginners set up a Claude Project router. Intermediate fork the build. Advanced map your own workflow.

> **We learned these this week. Now we'll show you what happened.**

DEMO · LEVEL 4

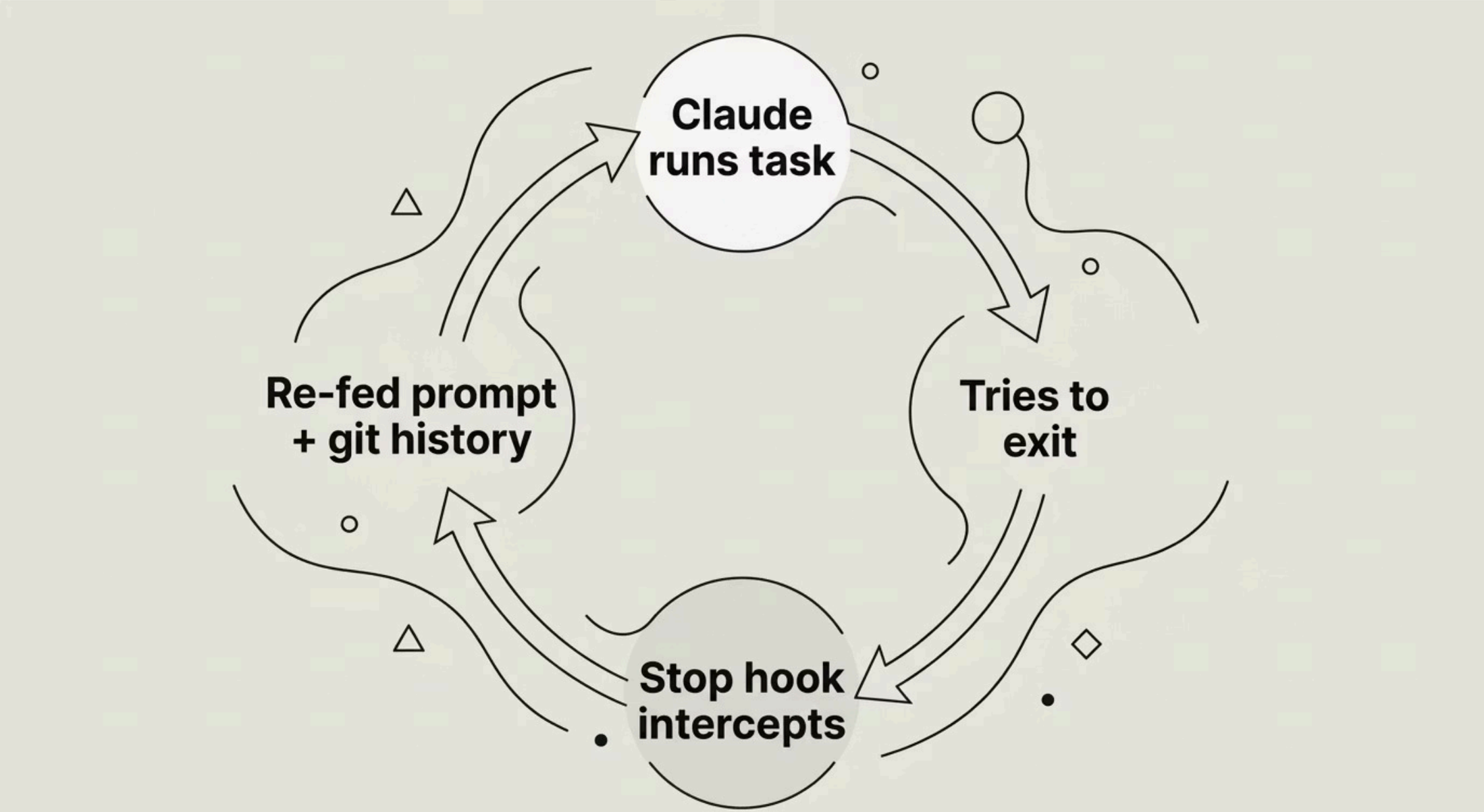
RALPH WIGGUM-LOUP

An autonomous loop that doesn't stop until the work is done.

DEMO · LEVEL 4

RALPH WIGGUM-LOUP

A Claude Code plugin that creates an autonomous loop. When Claude tries to exit, a Stop hook intercepts it and re-feeds the same prompt. Each iteration sees modified files and git history from all previous runs.



BACKPRESSURE BEATS MICROMANAGEMENT.

Tests and build failures create negative feedback loops — Claude reads the failure, fixes it, tries again.

HOW TO INSTALL & RUN

OFFICIAL PLUGIN

Built into Claude Code:

```
/ralph-loop "your task" --max-iterations 50 --completion-promise "DONE"
```

COMMUNITY VERSION

Extra safety features from frankbria:
github.com/frankbria/ralph-claude-code

BARE-BONES (HUNTLEY)

A bash while loop that pipes PROMPT.md into Claude Code. No dependencies. Just works.

KEY OPTIONS

--max-iterations	Always set this. Prevents runaway loops.
--completion-promise	Exact text Claude outputs when done.
--live	Stream output in real-time.
--resume <session_id>	Pick up where you left off.

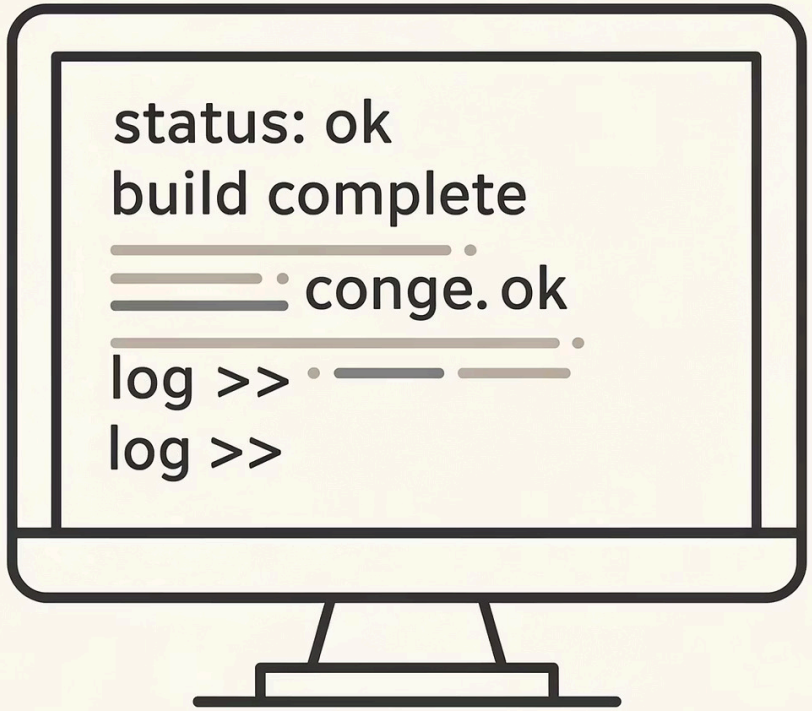
RALPH WIGGUM-LOUP

WHAT PEOPLE HAVE BUILT

> Geoffrey Huntley built an entire programming language — CURSED — with a working LLVM compiler. Over 3 months of autonomous runs.

> Another dev ran 27 hours straight, 84 tasks completed, building a Telegram bot. No human intervention.

The pattern: give it a hard problem, a test suite, and a way to know when it's done. Then get out of the way.



DEMO · LEVEL 5

KARPATHY'S AUTORESEARCH

Define a metric. Run experiments. Sleep. Wake up to results.

AUTORESEARCH

HOW IT WORKS

630 lines of Python. One tight loop.

1. DEFINE A METRIC

What does 'better' mean? Accuracy, speed, cost, score. Make it measurable.

2. RUN AN EXPERIMENT

Fixed 5-minute budget per run. Constrained, reproducible, comparable.

3. MEASURE THE OUTCOME

Did it improve? By how much? Log everything.

4. ANALYZE FAILURES

What went wrong? The agent reads its own results and diagnoses.

5. IMPROVE & REPEAT

Apply the fix. Run again. Loop until budget or goal is hit.

Karpathy pushed it March 7, went to sleep, woke up to 100+ experiments completed.

RALPH VS. AUTORESEARCH

TWO DIFFERENT PHILOSOPHIES OF AUTONOMOUS WORK.

RALPH WIGGUM-LOUP

- Open-ended iteration
- Task-driven: keep going until the work is done
- Feedback comes from tests, builds, and errors
- Best for: software projects, long coding tasks, anything with a test suite

> You define done. Claude figures out how.

KARPATHY'S AUTORESEARCH

- Structured eval loops
- Metric-driven: every experiment has a measurable outcome
- Feedback comes from scores and benchmarks
- Best for: optimization problems, ML experiments, any domain with a measurable metric

> You define better. Claude finds the path.

Key insight: AutoResearch isn't just for ML. Any domain with a measurable metric can use this pattern.

AUTORESEARCH

THE NUMBERS

700

EXPERIMENTS

in 2 days

20

OPTIMIZATIONS

discovered

19%

PERFORMANCE

gain (Shopify CEO, overnight run, 37 experiments)

25K+

GITHUB STARS

in 5 days

Claude Code port by uditgoenka generalizes the pattern beyond ML — any domain with a measurable metric. **github.com/uditgoenka/autoresearch**