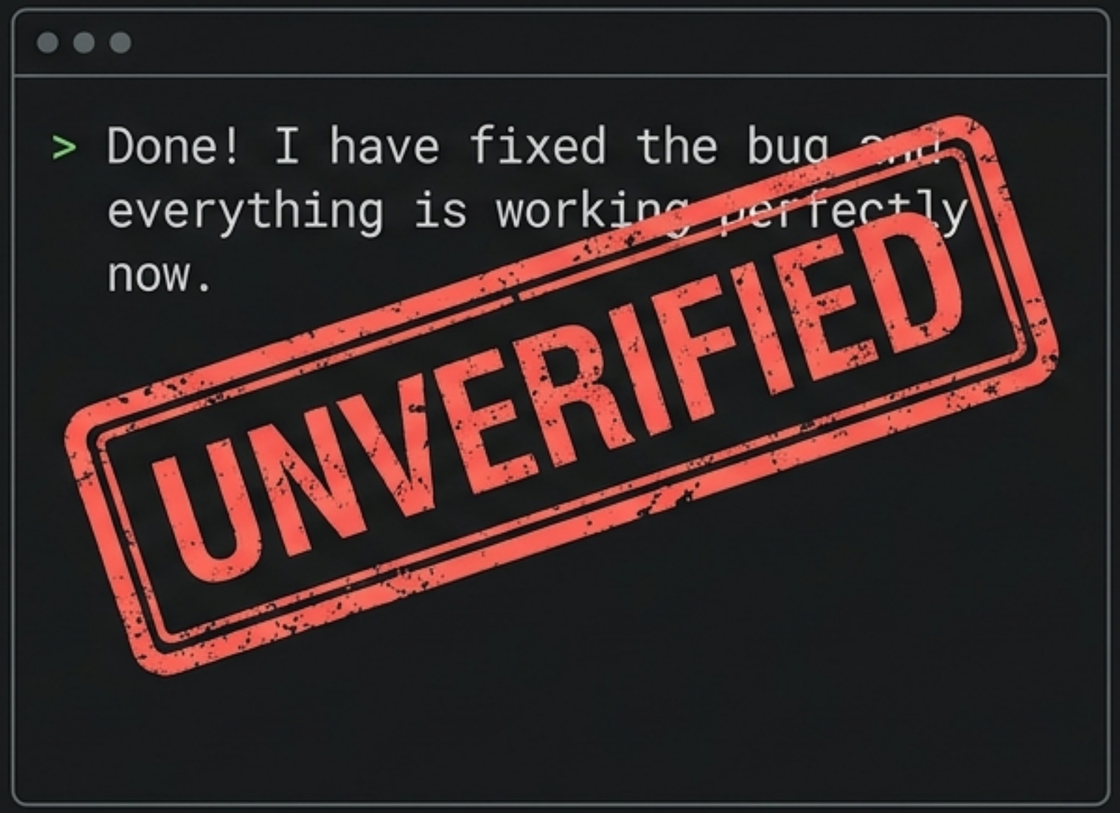
Enforcing the Burden of Proof

Introducing bulletproof-validation-gate for Claude Code

An adversarial stdlib-only Stop hook that intercepts unverified 'done' claims and forces the AI to prove them via a local Ollama model.

'Done' is Cheap. Proof is Not.



The Illusion of Completion

AI assistants routinely declare success without providing concrete, observable evidence.

The Manual QA Burden

The developer is forced to step in, run the commands, and verify the work by hand.

The Missing Constraint

Without enforcement, the system optimizes for conversational closure rather than verifiable reality.

The goal is not to trust the AI's claim.
The goal is to mathematically force it to provide the receipt.

The Adversarial Filter: Intercept, Disprove, Enforce

1. **Intercept**: Plugs directly into Claude Code as a Stop hook via `settings.json`.

2. **Route**: Redirects the final AI message and transcript tail to → a **local verification model.**

AI Final Message

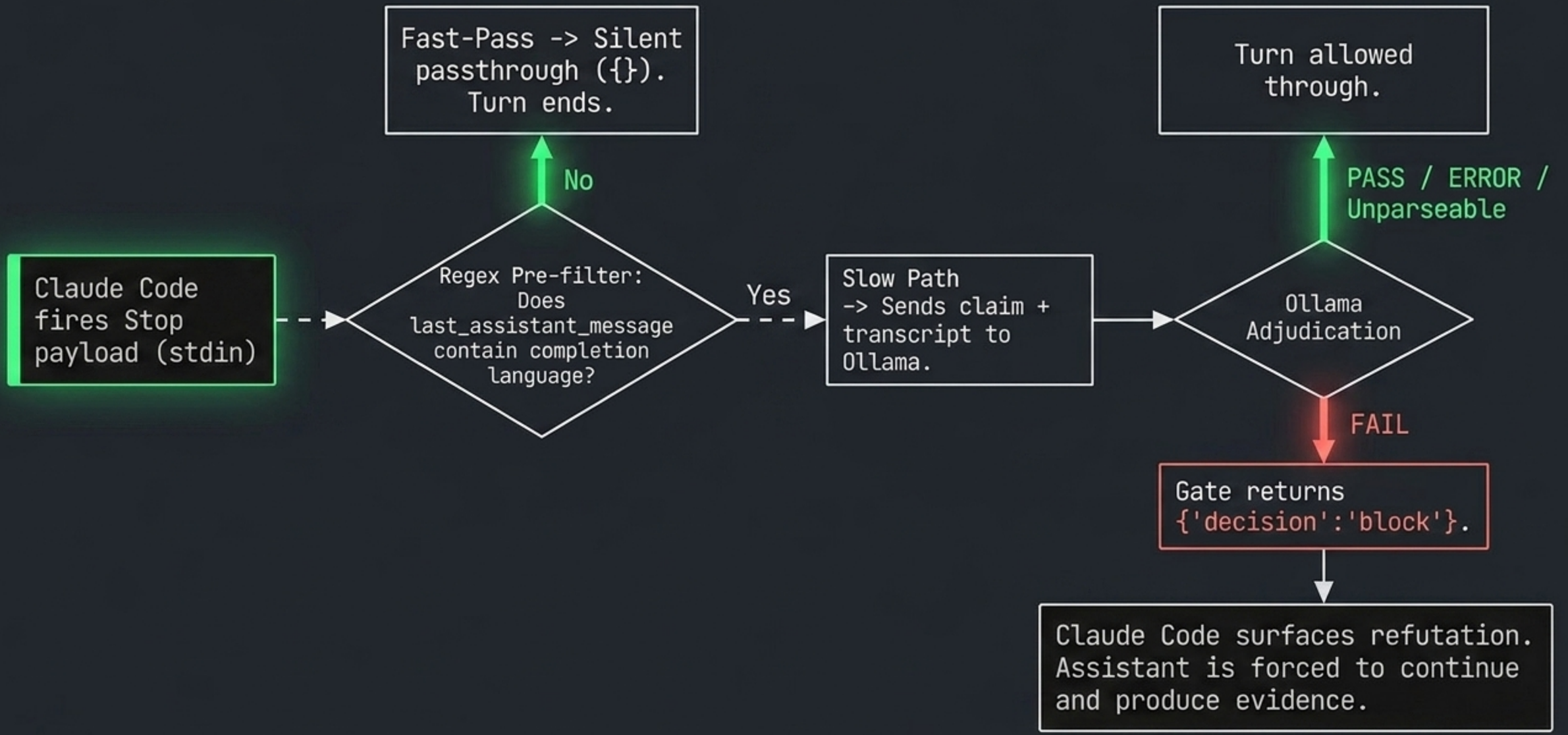
3. **Disprove**: Prompts the verifier adversarially—its only job is to find reasons the claim is unsupported.

4. **Enforce**: Returns a `{'decision':'block'}` if evidence is missing, forcing the AI to re-engage.

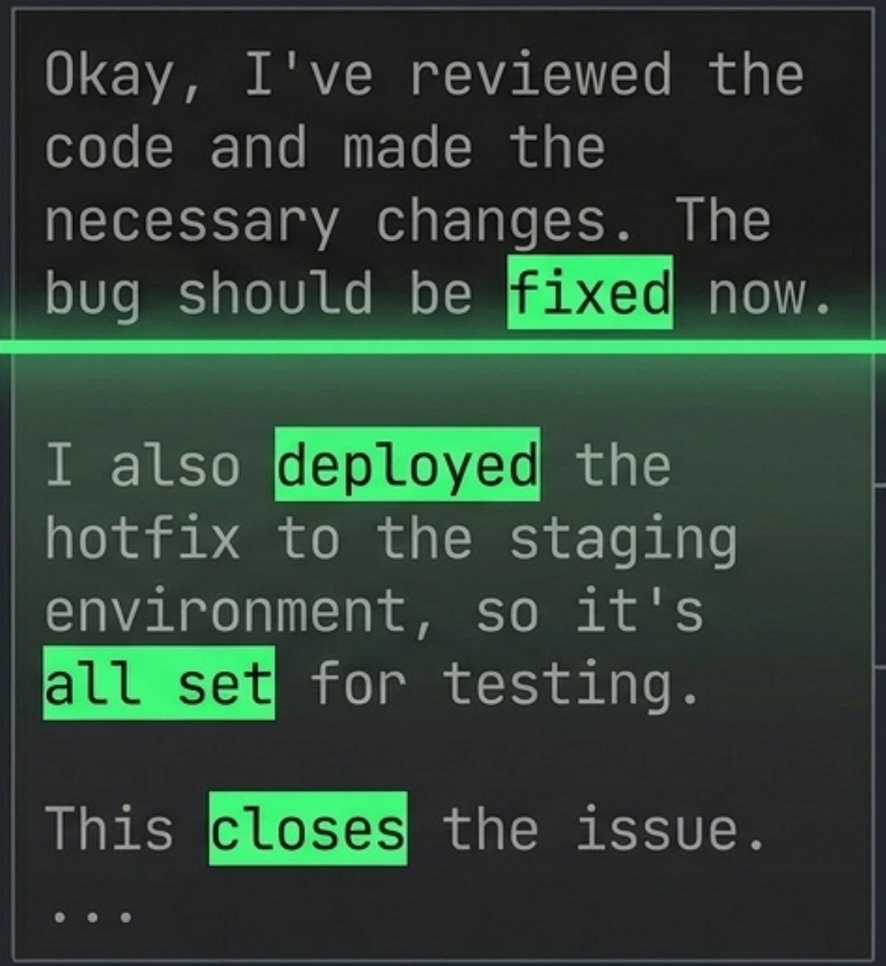
Stop Hook

```
>_
Entirely local via Ollama.
No API cost.
Nothing leaves the machine.
```

The Lifecycle of a Turn



The Fast Path: Cheap, Regex-Based Pre-Filtering



AI Message Scanner
(Regex Matcher)

Mechanism: Before invoking any models, the gate scans for case-insensitive, word-boundary completion patterns.

Trigger Vocabulary:

done | fixed | resolved | complete[d]
all set | working | deployed | in place
should resolve | this closes | up and running
good to go | ready to | verified

Performance Benefit: If the final message lacks these specific words, the verifier is never called. The turn hits the fast path and passes instantly.

-> Fast Pass (0ms latency)

The Evidence Contract: Proof vs. Promises

	❌ FAIL	✅ PASS
The Subject	Vague declarations ('Everything is fixed and working now').	Specific, testable subjects ('The /health endpoint returns 200').
The Proof	Describing what was done ('I wrote the code'). Future work described as complete.	Observable outputs (File paths, curl outputs, test results, timestamps).
The Alignment	Evidence provided does not match the actual claim made.	Evidence demonstrably proves the outcome OR claim explicitly downgraded ('pending verification', 'untested').

FAIL: 'Done. The login bug is fixed.'

PASS: 'Fixed login. curl -sI localhost:3000/login returns HTTP/1.1 200.'

The Adversarial Engine (Ollama Backend)

Single Endpoint

Makes one POST to OLLAMA_URL (default: `http://localhost:11434/api/generate`).

```
{
  'model': 'llama3.2:3b',
  'prompt': '...',
  'stream': false,
  'options':
    {'temperature': 0.1,
     'think': false,
     'keep_alive': '24h'}}
}
```

Backend-Agnostic

Swap models via OLLAMA_VERIFIER_MODEL (default: `llama3.2:3b`). Larger models judge more accurately; 3B models err toward strict blocking.

Stable Verdicts

Uses **temperature:** `0.1` for deterministic judgment.

Latency Optimized

Sets **think: `false`** to disable <think> preambles (defensively strips <think> and code fences if emitted), and uses **keep_alive: `24h`** to avoid cold-start latency.

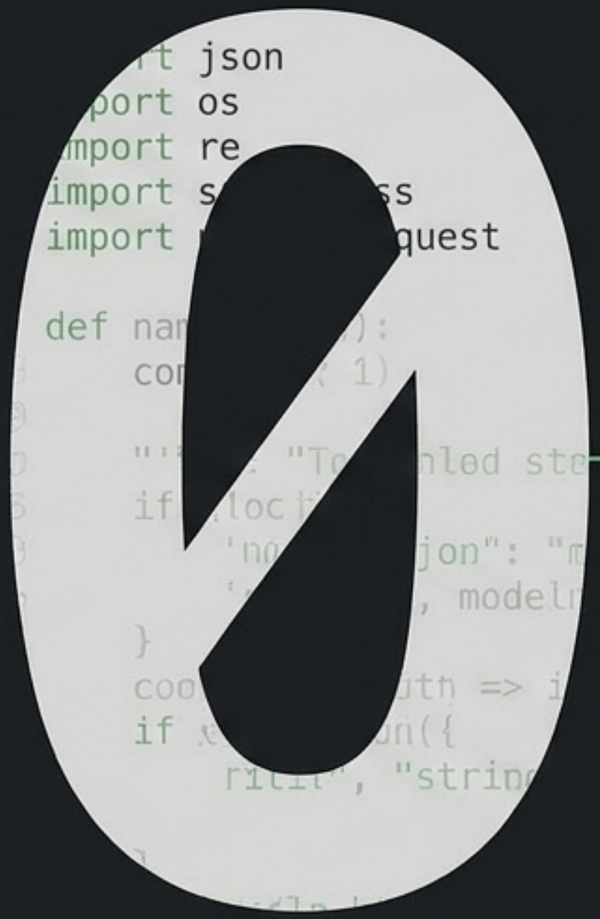
Designed to Fail-Open: Never Wedge the Session

A verification backend problem must never prevent a turn from finishing. The gate only enforces while Ollama is up.

1	Unreachable (OLLAMA_URL down, DNS error)	Verdict ERROR	PASS-THROUGH
2	Timeout (> OLLAMA_VERIFIER_TIMEOUT_S)	Verdict ERROR	PASS-THROUGH
3	Unparseable Model Output	Verdict UNPARSEABLE	PASS-THROUGH
4	Stop Payload Empty/Non-JSON		PASS-THROUGH
5	Loop Protection (stop_hook_active: true)	Prevents block loops	PASS-THROUGH

⚠ Takeaway: Only an explicit FAIL verdict produces a block. It degrades gracefully.

Zero Dependencies. Maximum Stability.



The Script ✅

completion_claim_gate.py is a single file importing only the Python standard library (CPython ≥ 3.8).

No Third-Party Code

0 distributable components. 0 runtime dependencies. No pip install, no virtualenv, no requirements.txt, no lockfiles to drift.

No Heavy Infrastructure

No Dockerfiles or compose stacks required.

Standard Library Utilization

Uses native json, os, re, subprocess, and urllib.request to handle logic and network calls natively.

Bottom Line: Built to run reliably inside a Claude Code Stop hook without breaking the user's environment.

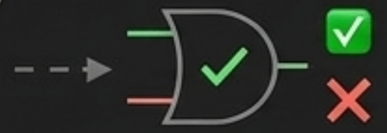
[structural parsing]

Operations: Tuning, Audit, and Alerts

Configuration (Environment Variables)

No restarts needed: Variables are read fresh on each Stop event.

1. `OLLAMA_VERIFIER_TIMEOUT_S` (Default: 20s) — Must sit below the `settings.json` hook timeout (35000ms).



2. `OLLAMA_VERIFIER_MODEL` — Tune up to larger models to reduce false blocks, or down for speed.

Audit & Notification

1. Verdict Logging: Appends to `VALIDATION_GATE_LOG` and writes full 2000-char excerpts to `VALIDATION_GATE_VERDICT_DIR` (`~/.claude/validation-gate/verdicts/`). Files are named `<UTC-timestamp>_<session-id>.json`.

2. `VALIDATION_GATE_NOTIFY`: An optional hook to execute a script on a `FAIL` verdict (`<script> error <subject> <body>`). Best-effort, 5s timeout.

Synthesis: Proof Over Promises