>_ Claude Code

Headroom for Claude Code

A free, open-source proxy that compresses logs, test output and JSON on your machine before Claude reads them. Two lines to set up. Checked against Headroom's repo and run once on October 13, 2026.

- **Logs & JSON · 90%+ fewer tokens***
- **Everyday coding · about 20%***

* Headroom's own numbers

Setup guide · October 2026

What Headroom does

1 It sits between Claude Code and the model.

A small proxy runs on your own machine. Claude Code talks to it, and it talks to Anthropic. "Compression runs on your machine; no prompt or file content is sent anywhere to be compressed" (README).

2 It shrinks what Claude reads.

Tool output, logs, files and JSON get compressed before they're sent. Repetitive output shrinks the most. The originals stay on your machine for a set time, and Claude can pull the full text back.

3 Your login and model stay the same.

We ran it on a Claude Max plan with no API key, and it worked. API users pay per token; plan users spend their limit. Either way, fewer tokens sent means less spent.

What it doesn't do: shrink short chats, prose, or output that's already tight. "Short conversational exchanges, prose, and already-dense payloads see little or no reduction" (README, When to skip).

Price: the open-source version is free (Apache-2.0). A paid Enterprise plan for teams exists. This guide uses only the free one.

Their numbers

From Headroom's README benchmark: four scenarios built from real MCP tool output.

SCENARIO	BEFORE → AFTER	SAVED
Code search 100 results	17,199 → 13,597	21%
Incident debugging logs from an outage	55,957 → 24,340	57%
Codebase exploration	58,801 → 33,895	42%
GitHub issue triage	46,067 → 32,429	30%
Log dump demo the GIF at the top of the README	10,144 → 1,260	88%

> "20% fewer tokens for coding agents, 60-95% fewer tokens for JSON, same answers."
>
> The repo's own description, github.com/headroomlabs-ai/headroom, October 13, 2026

On "same answers": its accuracy table shows the same GSM8K score (0.870) and no detectable difference on TruthfulQA. These are Headroom's tests, not ours.

Steps 1 · 2 · 3

The setup, in your terminal

```
HEADROOM FOR CLAUDE CODE: THE TWO-LINE SETUP

1. Install it (in your terminal)
pip install "headroom-ai[all]"
(Or keep it in its own environment: uv tool install --python
3.13 "headroom-ai[all]")
The npm package "headroom-ai" is a library only. The headroom
command comes from the pip package.

2. Start Claude Code through it
Instead of typing claude, type:
headroom wrap claude
Do this each time you start a session. It starts a small proxy
on your own machine and opens Claude Code routed through it.
Your Claude login and model stay the same.

3. Check that it works
headroom doctor       (health check, confirms the routing)
headroom dashboard    (live savings while a wrapped session is
running)
headroom savings      (the number for your own traffic)
```

Before you start

A few things to know

```
Good to know
- Biggest savings: long sessions with lots of logs, test
output and JSON. Short chats and prose barely shrink.
- The originals stay on your machine for a set time (a cache),
and Claude can pull the full text back during that time if it
needs it.
- It also sets up Serena, a code-navigation MCP server, for
that project. Skip it with: headroom wrap claude --code-memory
none
- An anonymous usage beacon is on by default (compression
ratios and counters, never prompts, code or file paths). Turn
it off with: export HEADROOM_BEACON=off
- Undo it with: headroom unwrap claude
```

Real run

What you'll see

The proxy starts, then Claude Code opens **routed through it**. Note the two lines it warns you about.

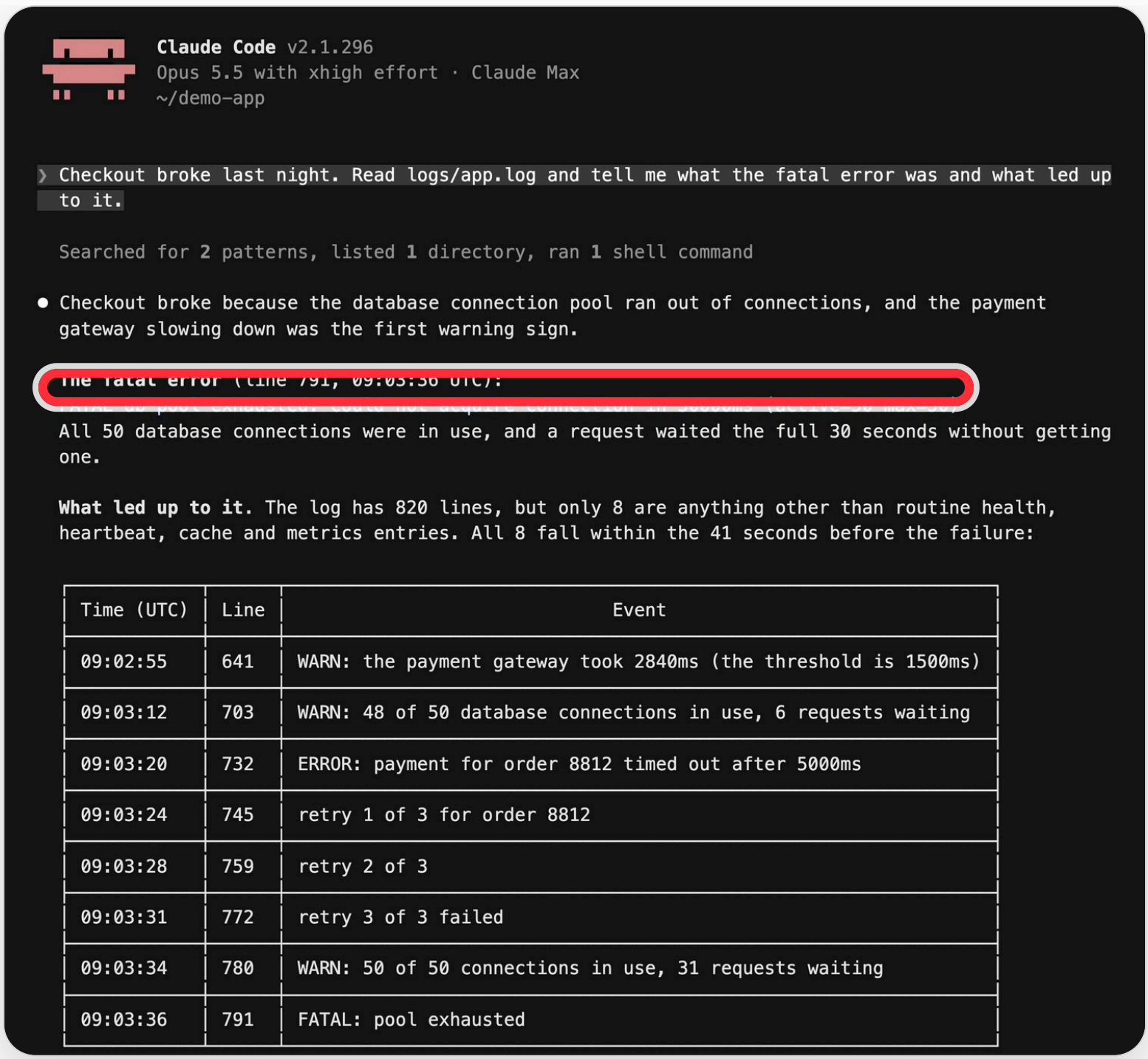
```
╔═══════════════════════════════════════════╗
║            HEADROOM WRAP: CLAUDE            ║
╚═══════════════════════════════════════════╝

  Starting Headroom proxy on port 8797...
  Proxy ready on http://127.0.0.1:8797
  Dashboard:    http://127.0.0.1:8797/dashboard

  Launching Claude Code (API routed through Headroom)...
  ANTHROPIC_BASE_URL=http://127.0.0.1:8797
  Remote Control: Claude Code 2.1.296 disables the /remote-control (/rc) command while
ANTHROPIC_BASE_URL points at a custom endpoint (the wrapped Claude session's ANTHROPIC_BASE_URL).
Headroom cannot override this client-side gate — run Claude without Headroom for sessions that
need Remote Control.
  Same base-URL gate also affects on-demand tool loading (#746 — Headroom keeps it on for this
session) and the 1M context window (#1158 — restore with `headroom wrap claude --1m`).

  ENABLE_TOOL_SEARCH=true (on-demand tool loading kept on; issue #746)
```

Remote Control (/rc) is off in a wrapped session, and the **1M context window** needs `headroom wrap claude --1m`. Both come from Claude Code turning those off behind a custom address, and Headroom says so on start.

Real run

Work as usual

We asked about an 820-line log with one fatal error in it. Claude found it and what led up to it.



Claude Code v2.1.296
Opus 5.5 with xhigh effort · Claude Max
~/demo-app

❯ Checkout broke last night. Read logs/app.log and tell me what the fatal error was and what led up to it.

Searched for **2** patterns, listed **1** directory, ran **1** shell command

● Checkout broke because the database connection pool ran out of connections, and the payment gateway slowing down was the first warning sign.

The fatal error (line 791, 09:03:36 UTC):

All 50 database connections were in use, and a request waited the full 30 seconds without getting one.

What led up to it. The log has 820 lines, but only 8 are anything other than routine health, heartbeat, cache and metrics entries. All 8 fall within the 41 seconds before the failure:

Time (UTC)	Line	Event
09:02:55	641	WARN: the payment gateway took 2840ms (the threshold is 1500ms)
09:03:12	703	WARN: 48 of 50 database connections in use, 6 requests waiting
09:03:20	732	ERROR: payment for order 8812 timed out after 5000ms
09:03:24	745	retry 1 of 3 for order 8812
09:03:28	759	retry 2 of 3
09:03:31	772	retry 3 of 3 failed
09:03:34	780	WARN: 50 of 50 connections in use, 31 requests waiting
09:03:36	791	FATAL: pool exhausted

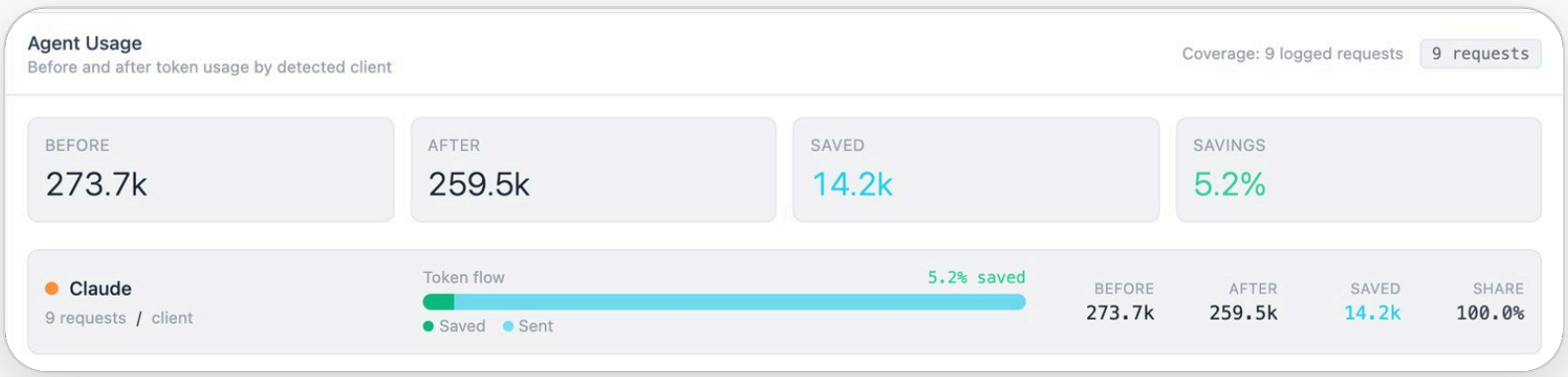
October 13, 2026. Claude Code 2.1.296, Max plan, Opus 5.5. We ran `headroom wrap claude --no-mcp --code-memory none` so our test machine's MCP settings stayed untouched. Only the folder path is shortened.

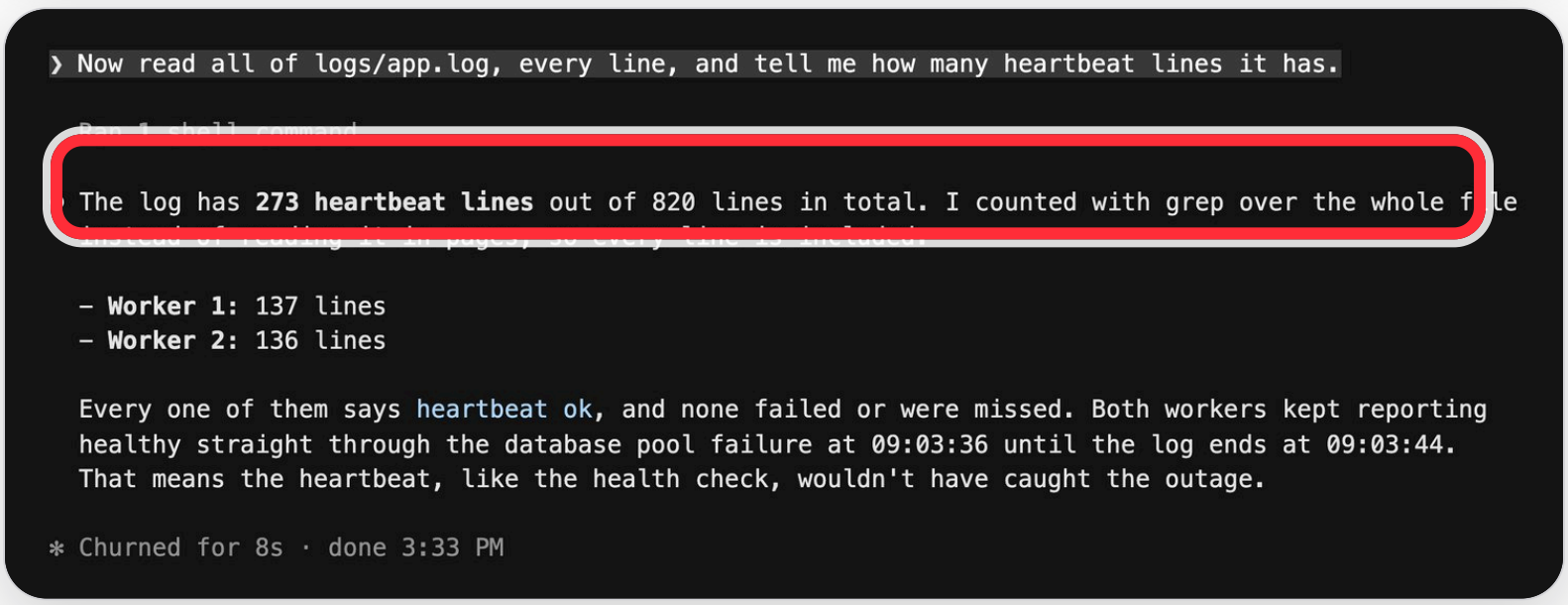
Real run

Our numbers, honestly

5.2%
message tokens saved across the session (273.7k → 259.5k)

2–8%
saved per request





Why so small: across both questions, Claude never dumped the log. It searched and read only the lines it needed, and counted with grep. So there was little junk to remove. Headroom pays off when raw output does reach the model: CI logs, big JSON from tools, database rows. Its dashboard also counted 103k "tool schema" tokens; we left those out, because Claude Code already loads tools on demand when it talks to Anthropic directly.

Questions

Is it slower?

A little. In our short run its dashboard showed about **1.25 seconds of overhead per request** on average. The README puts compression itself well under a millisecond on a 10K-token JSON result.

Do I type this every time?

Yes. Start sessions with **headroom wrap claude** instead of **claude**. Plain **claude** still works without it.

What does it change on my machine?

By default it registers MCP servers for Claude Code (its retrieve tool and Serena, a code-navigation server). Skip them with **--no-mcp** and **--code-memory none**. Undo with **headroom unwrap claude**.

Does it send data anywhere?

An anonymous beacon is on by default: compression ratios and counters, never prompts, code or file paths. Turn it off with **export HEADROOM_BEACON=off**.

pip install fails

The README's install section covers Rust and certificate errors. A prebuilt wheel: **pip install --only-binary headroom-ai headroom-ai**.

This DM came from an automation

You commented WRAP and Uplika sent this guide. Uplika lets Claude post to your channels, and sends a DM when someone comments the word you pick.

See how it works: uplika.com/en/integrations/claude

Headroom: `github.com/headroomlabs-ai/headroom`