Published April 16, 2026 · 33 min read · comparison

# Claude Code Statuslines Compared

A source-backed comparison of the Claude Code statusline ecosystem: frameworks, plugins, shell scripts, Rust binaries, Ghostty-friendly quick-glance layouts, and the platform gaps still shaping the market.

comparison developer-tools

Date: 2026-04-16


## Executive Summary

- There is no single best Claude Code statusline anymore. There are at least five distinct product shapes:
  - a configurable framework: ccstatusline (https://github.com/sirmalloc/ccstatusline)
  - a plugin-first themeable powerline: claude-powerline (https://github.com/Owloops/claude-powerline)
  - a Rust performance play: CCometixLine (https://github.com/Haleclipse/CCometixLine)
  - a Starship bridge: cship (https://github.com/stephenleo/cship)

    - a transcript-aware operational HUD: claude-hud (https://github.com/jarrodwatts/claude-hud)

- ccstatusline (https://github.com/sirmalloc/ccstatusline) remains the category leader because it is the most legible general-purpose answer for most people. It has the biggest adoption signal in the dedicated-statusline field, the broadest widget surface, and the most approachable TUI editor.
- The ecosystem is now differentiated less by "can it show model, context, git, and cost?" and more by:
    - how it installs
    - what data it trusts
    - whether it performs network or transcript work at render time
    - how much terminal/aesthetic ambition it carries
- Official Claude Code support is now good enough that a lot of older hackiness is no longer mandatory. The `statusLine` payload already carries model, context, cost, worktree, agent, and `rate_limits`. But the open issue queue shows the platform is still missing several fields and refresh guarantees that builders clearly want.
- The best quick-glance layout I tested in day-to-day use is still not a framework. It is the shell lineage closest to SippieCup's March 30, 2026 gist (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871), especially when tuned for Ghostty with stronger severity colors and a giant context row.

> *"Short answer: If you want one recommendation without overthinking it, use ccstatusline (https://github.com/sirmalloc/ccstatusline). If you want the cleanest plugin-native path, use claude-powerline (https://github.com/Owloops/claude-powerline). If you care more about what Claude is doing than about pure visual polish, use claude-hud (https://github.com/jarrodwatts/claude-hud)."*

-

## Inclusion Criteria and Market Definition

This report includes public GitHub-hosted projects that meet at least one of these conditions:

- they are explicitly built as a Claude Code statusline
- they install into Claude Code's `statusLine` hook as a first-class feature
- they are statusline-adjacent enough that users genuinely compare them in practice, such as ccusage (https://github.com/ryoppippi/ccusage)'s beta `statusline` subcommand or claude-hud (https://github.com/jarrodwatts/claude-hud)'s HUD-style status area

This report does not try to catalog every personal `~/.claude/statusline.sh` dotfile in existence. It is focused on projects that have public docs, public install guidance, or enough community visibility to matter as an option category.

## Market Map

-

ccstatusline (https://github.com/sirmalloc/ccstatusline) claude-powerline (https://github.com/Owloops/claude-powerline) CCometixLine (https://github.com/Haleclipse/CCometixLine) cship (https://github.com/stephenleo/cship)

claude-hud (https://github.com/jarrodwatts/claude-hud) claudeline (https://github.com/fredrikaverpil/claudeline)

kamranahmedse/claude-statusline (https://github.com/kamranahmedse/claude-statusline) daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine) felipeelias/claude-statusline (https://github.com/felipeelias/claude-statusline) claude-pace (https://github.com/Astro-Han/claude-pace) kcchien/claude-code-statusline (https://github.com/kcchien/claude-code-statusline) pyccsl (https://github.com/wolfdenpublishing/pyccsl) chongdashu/cc-statusline (https://github.com/chongdashu/cc-statusline)

pcvelz/ccstatusline-usage (https://github.com/pcvelz/ccstatusline-usage) syou6162/ccstatusline (https://github.com/syou6162/ccstatusline) FlineDev/CustomStatusline (https://github.com/FlineDev/CustomStatusline) ohugonnot/claude-code-statusline (https://github.com/ohugonnot/claude-code-statusline) xleddyl/claude-watch (https://github.com/xleddyl/claude-watch) rz1989s/claude-code-statusline (https://github.com/rz1989s/claude-code-statusline) Wangnov/claude-code-statusline-pro (https://github.com/Wangnov/claude-code-statusline-pro) sotayamashita/claude-code-statusline (https://github.com/sotayamashita/claude-code-statusline) daliovic/cc-statusline (https://github.com/daliovic/cc-statusline) leeguooooo/claude-code-usage-bar (https://github.com/leeguooooo/claude-code-usage-bar)

-

## What Changed Since the Earliest Statusline Wave

The earliest public statuslines were mostly shell scripts compensating for missing platform data. The current market looks different because the official Claude Code payload now exposes much more:

- `model`
- `cost`
- `context_window`
- `agent`
- `worktree`
- `rate_limits`
- multi-line ANSI output and OSC 8 links

That changed the job. The most interesting statusline tools in 2026 are no longer just "parsing stdin nicely." They are making product decisions around configuration UX, extra data sources, security posture, and visual hierarchy.

## Official Claude Code Statusline Platform Baseline

The official Claude Code statusline docs (https://docs.anthropic.com/en/docs/claude-code/statusline) define a very simple contract:

- Claude Code runs a shell command from `settings.json`
- it sends a JSON blob over stdin
- your command prints a string or multi-line block
- ANSI colors and OSC 8 links are supported

That simplicity is why the ecosystem exploded so fast.

-

## The Baseline Capability Set

As of April 16, 2026, the official payload is already rich enough for most mainstream statuslines:

- model identity via `model.id` and `model.display_name`
- session cost and code-change stats via `cost.*`
- context-window size and current usage via `context_window.*`
- rate-limit windows via `rate_limits.five_hour` and `rate_limits.seven_day`
- workspace and worktree context
- agent name
- output style and vim mode

There are also two important product-level conveniences:

- `/statusline` can scaffold a statusline from a natural-language prompt
- the official docs support multi-line output, which is why two-line and three-line designs are now common instead of hacks

## The Practical Consequence

If you only need model, context, cost, and subscriber limits, you no longer need to scrape credentials or parse logs. The builders who still do extra work are usually trying to get one of four things:

- more accurate or broader usage tracking
- live tool or subagent activity
- richer session analytics like cache efficiency or burn rate
- better install and configuration ergonomics

-

## Comparison Framework and Signal Legend

This report compares projects across five lenses.

```
stdin
```

Signal labels used below:

- `Leader`: strong adoption plus a differentiated product shape
- `Specialist`: clear value for a specific user or workflow
- `Appendix`: interesting, but narrower, newer, smaller, or too overlapping to justify core-tier treatment

## Tier 1: Frameworks, Plugins, and Full Systems

## Core Comparison

-

| | | | |
|---|---|---|---|
| sirmalloc/ccstatusline (https://github.com/sirmalloc/ccstatusline) | | | `npx` `bunx` |
| Owloops/claude-powerline (https://github.com/Owloops/claude-powerline) | | | `npx` |
| Haleclipse/CCometixLine (https://github.com/Haleclipse/CCometixLine) | | | |
| stephenleo/cship (https://github.com/stephenleo/cship) | | | `cargo install` |
| jarrodwatts/claude-hud (https://github.com/jarrodwatts/claude-hud) | | | |
| fredrikaverpil/claudeline (https://github.com/fredrikaverpil/claudeline) | | | `go install` |

## Capability Matrix

-

| | CCSTATUSLINE (HTTPS://GITHUB.COM/SIRMALLOC/CCSTATUSLINE) | CLAUDE-POWERLINE |
| --- | --- | --- |

## What Each Tier-One Leader Actually Wins

### ccstatusline (https://github.com/sirmalloc/ccstatusline)

ccstatusline (https://github.com/sirmalloc/ccstatusline) is still the category leader because it looks and behaves like a real product instead of a script pack. It has the broadest widget system, the best-known TUI editor, multi-line flexibility, powerline support, and a genuinely approachable default experience for users who do not want to hand-author TOML or shell.

Its main weakness is not capability. It is operational posture. The easiest documented path still leans on moving-version `npx` or `bunx` patterns, and the project surface can feel larger than some users really need.

-

## claude-powerline (https://github.com/Owloops/claude-powerline)

If ccstatusline (https://github.com/sirmalloc/ccstatusline) is the framework answer, claude-powerline (https://github.com/Owloops/claude-powerline) is the plugin-native answer. It feels like a polished Claude plugin first and a statusline package second: slash-command setup, config auto-reload, theme families, Unicode or ASCII modes, and a visual taste that is much more deliberate than most utilitarian lines.

For users who want something pretty and strongly packaged without living in a TUI, this is the cleanest recommendation.

## CCometixLine (https://github.com/Haleclipse/CCometixLine)

CCometixLine (https://github.com/Haleclipse/CCometixLine) is not just "Rust ccstatusline (https://github.com/sirmalloc/ccstatusline)." It has a different personality. The project combines a Rust statusline binary, built-in themes, a TUI, transcript-based usage logic, and optional Claude Code patching utilities like context-warning disabling and verbose-mode helpers.

That makes it powerful, but also broader and more invasive than a pure formatter. People who want a strict statusline-only tool may love the rendering speed and dislike the adjacent utility surface.

## cship (https://github.com/stephenleo/cship)

cship (https://github.com/stephenleo/cship) has the cleanest technical thesis in the whole market: reuse the Starship mental model instead of inventing another bespoke config language. If you already invested in `starship.toml`, this is the only serious option that lets that investment follow you into Claude Code.

The tradeoff is obvious too. If you do not care about Starship, some of cship (https://github.com/stephenleo/cship)'s brilliance is wasted on you.

## claude-hud (https://github.com/jarrodwatts/claude-hud)

claude-hud (https://github.com/jarrodwatts/claude-hud) is arguably not a statusline in the classic sense anymore. It is a compact HUD. The reason it matters is simple: the official stdin payload still does not tell you what tool Claude is currently using, which subagents

are alive, or how the todo list is moving. claude-hud (https://github.com/jarrodwatts/claude-hud) gets that by reading transcript JSONL, and that makes it the strongest answer for "what is Claude doing right now?"

If your priority is operational awareness rather than visual minimalism, claude-hud (https://github.com/jarrodwatts/claude-hud) has the clearest product value in the space.

### claudeline (https://github.com/fredrikaverpil/claudeline)

claudeline (https://github.com/fredrikaverpil/claudeline) is a small project with a much stronger architecture story than its star count implies. It is a Go stdlib binary, has an offline capture/render workflow, distinguishes subscription plan versus provider, exposes service disruption state, and thinks carefully about cache TTLs and failure behavior.

This is the project I would point infrastructure-minded users toward when they want something intentionally constrained and operationally legible.

## Tier-One Verdict

- Best general default: ccstatusline (https://github.com/sirmalloc/ccstatusline)
- Best plugin-first setup: claude-powerline (https://github.com/Owloops/claude-powerline)
- Best Rust product surface: CCometixLine (https://github.com/Haleclipse/CCometixLine)
- Best for existing Starship users: cship (https://github.com/stephenleo/cship)
- Best observability layer: claude-hud (https://github.com/jarrodwatts/claude-hud)
- Best minimal binary with strong operational thinking: claudeline (https://github.com/fredrikaverpil/claudeline)

## Tier 2: Opinionated Midweights

These projects do not define the whole market, but they are exactly where most of the interesting taste lives.

-

| | | | |
|---|---|---|---|
| kamranahmedse/claude-statusline (https://github.com/kamranahmedse/claude-statusline) | | | `~/.claude/` |
| daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine) | | | |
| felipeelias/claude-statusline (https://github.com/felipeelias/claude-statusline) | | | |
| Astro-Han/claude-pace (https://github.com/Astro-Han/claude-pace) | | | |
| kcchien/claude-code-statusline (https://github.com/kcchien/claude-code-statusline) | | | |
| wolfdenpublishing/pyccsl (https://github.com/wolfdenpublishing/pyccsl) | | | |
| chongdashu/cc-statusline (https://github.com/chongdashu/cc-statusline) | | | |

-

leeguooooo/claude-code-usage-bar (https://github.com/leeguooooo/claude-code-usage-bar)

## The Midweights That Matter Most

### kamranahmedse/claude-statusline (https://github.com/kamranahmedse/claude-statusline)

This is the "I just want a sane default from someone I already trust" option. It is small, opinionated, and does the important boring things correctly: copy a script, patch settings, back up what existed before, and offer uninstall.

That is a more valuable product shape than people sometimes give it credit for.

### daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine)

Daniel Oliveira's repo is one of the strongest cross-platform reference implementations in the public set. The Bash and PowerShell pairing matters because a lot of otherwise good statusline repos quietly stop being serious the moment Windows enters the discussion.

It also makes the old "copy this script and let Claude install it for me" workflow extremely legible.

### felipeelias/claude-statusline (https://github.com/felipeelias/claude-statusline)

Felipe's project is tiny in adoption but conceptually clean: Go, TOML, presets, preview commands, and hyperlink support. It feels like a prompt engine written by someone who wanted a statusline in Go because none of the existing tools matched that preference.

I would not call it market-leading yet, but it is one of the better-structured small projects.

-

### claude-pace (https://github.com/Astro-Han/claude-pace)

claude-pace (https://github.com/Astro-Han/claude-pace) deserves more attention than its star count implies because it reframes the problem correctly. A raw "60% used" number is weak information. A pace delta that tells you whether you are outrunning the remaining window is much more operationally useful.

If you care mostly about quota behavior and want pure Bash plus `jq`, this is one of the smartest narrow tools in the whole category.

### kcchien/claude-code-statusline (https://github.com/kcchien/claude-code-statusline)

This repo matters as a design reference even if it never becomes the biggest project in the field. The gradient context bar, truecolor fallback story, smart hiding, and compact density are all good examples of statusline design that cares about peripheral readability instead of just stuffing in more tokens.

It is one of the strongest public examples of "dense, shell-native, and still tasteful."

### pyccsl (https://github.com/wolfdenpublishing/pyccsl)

pyccsl (https://github.com/wolfdenpublishing/pyccsl) is the best answer for someone who wants a zero-dependency Python statusline with unusually rich metrics. Cache hit rate, token breakdowns, response timing, and theme variety make it one of the most information-rich non-framework tools available.

That said, it is still a single-file Python world. That is a strength for some users and a hard no for others.

## Tier 3: Shell Scripts, Specialists, and Long-Tail Projects

The long tail matters because this market still innovates through shell scripts faster than polished frameworks do.

-

## The Long-Tail Projects Worth Knowing

- xleddyl/claude-watch (https://github.com/xleddyl/claude-watch)
  - Important not because it is huge, but because it normalized the "refresh the usage cache in hooks, keep render fast" pattern.
- FlineDev/CustomStatusline (https://github.com/FlineDev/CustomStatusline)
  - Pluginized rate-limit monitor. Good reference if your main concern is usage windows, not a whole statusline framework.
- ohugonnot/claude-code-statusline (https://github.com/ohugonnot/claude-code-statusline)
  - A very direct script-first answer for `/oauth/usage` tracking and reset countdowns.
- syou6162/ccstatusline (https://github.com/syou6162/ccstatusline)
  - Despite the name collision, this is a separate Go project with a different philosophy: YAML-defined shell actions and caches rather than fixed widgets.
- sotayamashita/claude-code-statusline (https://github.com/sotayamashita/claude-code-statusline)
  - A small Rust project borrowing Starship's modular ideas into an embeddable Claude-specific binary.
- Wangnov/claude-code-statusline-pro (https://github.com/Wangnov/claude-code-statusline-pro)
  - Multilingual, preset-heavy, and much more ambitious than a simple shell script, but not yet a category-shaping choice outside its current audience.
- rz1989s/claude-code-statusline (https://github.com/rz1989s/claude-code-statusline)
  - Extremely feature-heavy shell suite with many components and installer transparency language. It is interesting, but feature maximalism is not the same thing as category leadership.
- daliovic/cc-statusline (https://github.com/daliovic/cc-statusline)

- Distinctive for prayer times and MCP-oriented extras.

## Why the Long Tail Still Matters

The long tail keeps discovering patterns that later frameworks copy:

- hook-based background refreshes
- better SSH or fallback glyph stories
- more truthful rate-limit semantics
- smarter context bars
- novel install patterns

The frameworks get the stars. The scripts still do a lot of the original invention.

## Architecture Patterns and Data-Source Lineages

This is the single most useful way to understand the market.

-

| | | | |
|---|---|---|---|
| | claude-powerline (https://github.com/Owloops/claude-powerline) felipeelias/claude-statusline (https://github.com/felipeelias/claude-statusline) claude-pace (https://github.com/Astro-Han/claude-pace) | | |
| | claudeline (https://github.com/fredrikaverpil/claudeline) daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine) | `/api/oauth/usage` | |
| | claude-hud (https://github.com/jarrodwatts/claude-hud) CCometixLine (https://github.com/Haleclipse/CCometixLine) pyccsl (https://github.com/wolfdenpublishing/pyccsl) | | |
| | xleddyl/claude-watch (https://github.com/xleddyl/claude-watch) | | |
| | chongdashu/cc-statusline (https://github.com/chongdashu/cc-statusline) kamranahmedse/claude-statusline (https://github.com/kamranahmedse/claude-statusline) | | |
| | claude-powerline (https://github.com/Owloops/claude-powerline) claude-hud (https://github.com/jarrodwatts/claude-hud) claudeline (https://github.com/fredrikaverpil/claudeline) claude-pace (https://github.com/Astro-Han/claude-pace) | | |

-

## The Biggest Lineage Split

The most important historical split is this:

- pre-`rate_limits` statuslines often scraped OAuth credentials and queried Anthropic directly
- post-`rate_limits` statuslines increasingly prefer official stdin data and only do extra work for features the payload still lacks

That is why modern statuslines look more like product choices than hacks. The baseline is now stable enough that extra complexity usually reflects deliberate ambition, not pure necessity.

## Quick-Glance Shell Lineage: The Ghostty Case Study

After trying more than ten public statusline variants, the most visually satisfying one I used in practice was still a tuned shell script, not a framework.



## Why This Layout Works So Well

- it is legible from peripheral vision, not just from close reading
- the current and weekly windows sit on the same line, so the relationship is obvious
- the colors shift aggressively enough that you notice them instantly in Ghostty
- the giant third-line context row turns "how full am I?" into a shape rather than a number
- the line feels calm even when it is information-dense

-

That combination is still surprisingly rare.

> *“Field note: I have tried more than ten public Claude Code statusline variants at this point. For quick visual satisfaction, especially in Ghostty, this is still the one I most enjoy glancing at.”*

## Closest Public Upstream Found

The closest public upstream I found for the local script lineage is SippieCup's March 30, 2026 gist (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871).

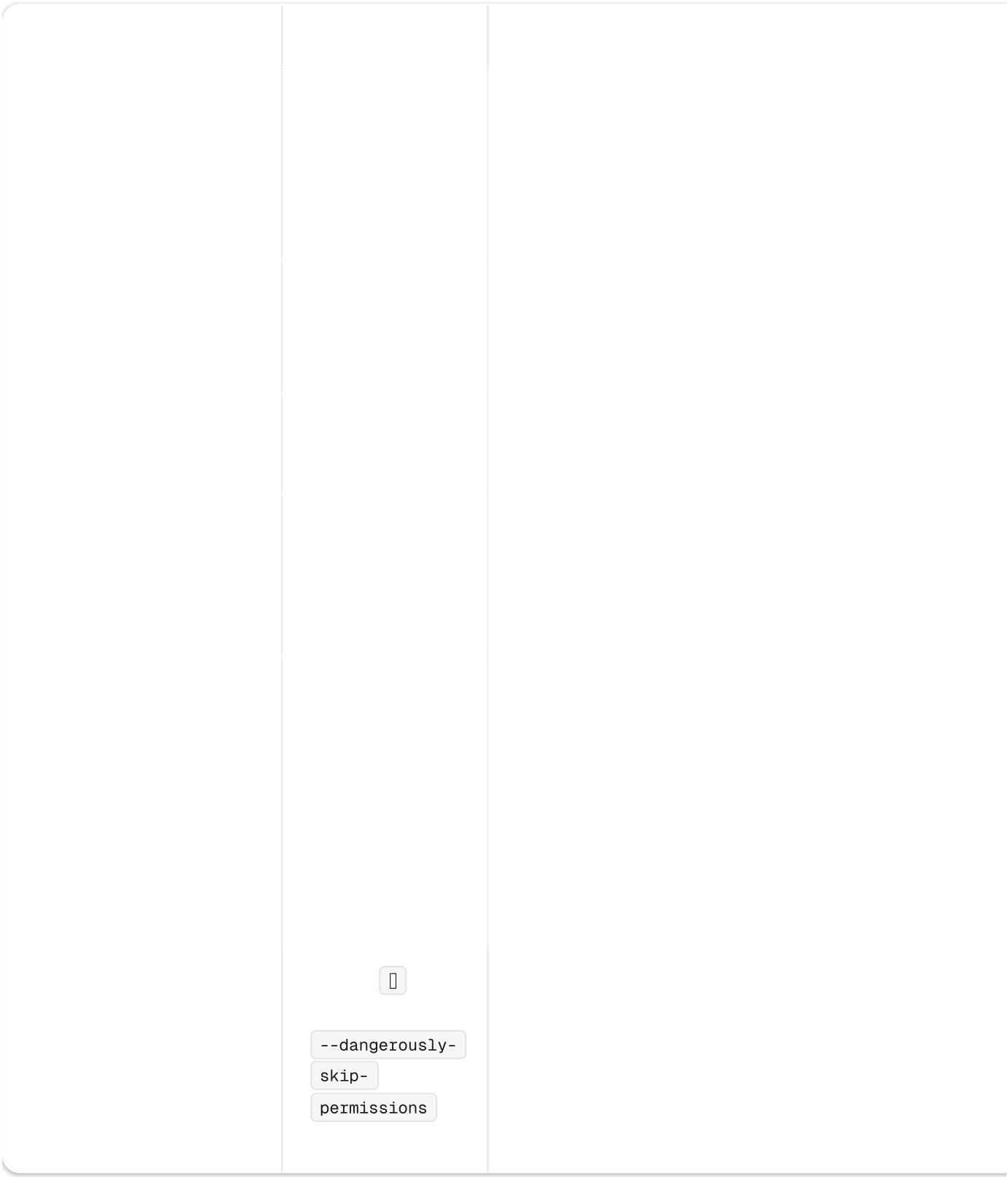
That is the strongest public code overlap I found. There are earlier public relatives in the same family, especially jtbr's February 8, 2026 gist (https://gist.github.com/jtbr/4f99671d1cee06b44106456958caba8b), but the local script overlaps much more strongly with SippieCup (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871) than with the earlier public gist chain.

Important nuance:

- I am not claiming a perfect one-hop provenance chain
- I am saying SippieCup (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871) is the closest public upstream I could verify from the code

## What Changed in the Local Ghostty-Tuned Variant

-




This is the best example I found of a statusline that optimizes for "observable in one glance" instead of "pack in one more metric."

-

## Security, Reliability, and Supply-Chain Risks

The biggest ecosystem gap is not missing features. It is update trust.

## The Core Risk

Several popular projects still document install paths like:

- `npx -y ccstatusline@latest`
- `npx -y @owloops/claude-powerline@latest --style=powerline`
- `npx claude-pace`

Combined with the official model where Claude Code reruns the statusline command repeatedly, that creates an obvious operational question:

- do you want a constantly re-executed UI hook to resolve moving package versions at runtime?

That risk statement is an inference from the official hook contract plus the documented install patterns, not a claim of a known compromise. But the operational tradeoff is still real.

## Reliability Matrix

-

| | | |
|---|---|---|
| npx | claudeline (https://github.com/fredrikaverpil/claudeline) CCometixLine (https://github.com/Haleclipse/CCometixLine) cship (https://github.com/stephenleo/cship) kamranahmedse/claude-statusline (https://github.com/kamranahmedse/claude-statusline) daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine)<br><br>ccstatusline (https://github.com/sirmalloc/ccstatusline) claude-powerline (https://github.com/Owloops/claude-powerline) claude-pace (https://github.com/Astro-Han/claude-pace) npx<br><br>claudeline (https://github.com/fredrikaverpil/claudeline) daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine) ohugonnot/claude-code-statusline (https://github.com/ohugonnot/claude-code-statusline)<br><br>claude-hud (https://github.com/jarrodwatts/claude-hud) CCometixLine (https://github.com/Haleclipse/CCometixLine) pyccsl (https://github.com/wolfdenpublishing/pyccsl)<br><br>xleddyl/claude-watch (https://github.com/xleddyl/claude-watch) | |

-

## Practical Guidance

- If you want the lowest-drama setup, prefer a copied script or installed binary over `npx @latest` in the hot path.
- If you want richer subscriber-limit data, cached API polling is still reasonable, but only if the tool handles credential lookup and cache TTLs sanely.
- If you want tool and subagent visibility, transcript parsing is still the only serious path, but you should accept that it is inherently closer to Claude Code's evolving internals.

## Recommendations by Persona and Use Case

-

ccstatusline (https://github.com/sirmalloc/ccstatusline)

claude-powerline (https://github.com/Owloops/claude-powerline)

cship (https://github.com/stephenleo/cship)

CCometixLine (https://github.com/Haleclipse/CCometixLine)

claude-hud (https://github.com/jarrodwatts/claude-hud)

claudeline (https://github.com/fredrikaverpil/claudeline)

claude-pace (https://github.com/Astro-Han/claude-pace)

daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine)

pyccsl (https://github.com/wolfdenpublishing/pyccsl)

-

| | | |
|---|---|---|
| | kcchien/claude-code-statusline (https://github.com/kcchien/claude-code-statusline) | |
| | SippieCup (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871) | |

## Open Platform Gaps and GitHub Issue Watchlist

The open anthropics/claude-code (https://github.com/anthropics/claude-code) queue makes the market's pressure points pretty clear. As of April 16, 2026, these are the issues I would watch most closely.

-

`#48445` (https://github.com/anthropics/claude-code/issues/48445) `statusLine.refreshInterval`

`#47071` (https://github.com/anthropics/claude-code/issues/47071)

`#49022` (https://github.com/anthropics/claude-code/issues/49022) `context_breakdown`

`#49270` (https://github.com/anthropics/claude-code/issues/49270)

`#47534` (https://github.com/anthropics/claude-code/issues/47534)

`#44982` (https://github.com/anthropics/claude-code/issues/44982)

`#40279` (https://github.com/anthropics/claude-code/issues/40279)

`#40287` (https://github.com/anthropics/claude-code/issues/40287) `/rename`

`#37216` (https://github.com/anthropics/claude-code/issues/37216) `tmux`

The meta-pattern is straightforward:

- builders want better repaint semantics
- builders want a few more operational fields

- builders want Unicode and hyperlink behavior to be more predictable across terminals

## Methodology and Full Source Appendix

## Methodology

- Research date: April 16, 2026
- Primary source order:
  - official Claude Code docs
  - current GitHub repo metadata via `gh`
  - current project READMEs
  - open anthropics/claude-code (https://github.com/anthropics/claude-code) issues
  - local lineage comparison against public gists
- Coverage:
  - 24 public GitHub projects with statusline or statusline-capable relevance
  - 2 public gists used for lineage analysis
  - current repo stars, push dates, release tags, and install paths checked live on publication day

## Appendix Catalog: Notable Projects Outside the Main Narrative

-

pcvelz/ccstatusline-usage (https://github.com/pcvelz/ccstatusline-usage)

ccstatusline (https://github.com/sirmalloc/ccstatusline)

FlineDev/CustomStatusline (https://github.com/FlineDev/CustomStatusline)

ohugonnot/claude-code-statusline (https://github.com/ohugonnot/claude-code-statusline)

`/oauth/usage`

xleddyl/claude-watch (https://github.com/xleddyl/claude-watch)

syou6162/ccstatusline (https://github.com/syou6162/ccstatusline)

daliovic/cc-statusline (https://github.com/daliovic/cc-statusline)

rz1989s/claude-code-statusline (https://github.com/rz1989s/claude-code-statusline)

Wangnov/claude-code-statusline-pro (https://github.com/Wangnov/claude-code-statusline-pro)

sotayamashita/claude-code-statusline (https://github.com/sotayamashita/claude-code-statusline)

ryoppippi/ccusage (https://github.com/ryoppippi/ccusage)

`npx ccusage statusline`

leeguooooo/claude-code-usage-bar (https://github.com/leeguooooo/clau

| | | |
|---|---|---|
| de-code-usage-bar) | | |
| chongdashu/cc-statusline (https://github.com/chongdashu/cc-statusline) | | |

## Notable Gists

| | | |
|---|---|---|
| SippieCup/0cd2567... (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871) | | |
| jtbr/4f99671... (https://gist.github.com/jtbr/4f99671d1cee06b44106456958caba8b) | | |

## Bottom Line

The Claude Code statusline market is mature enough now that "best" is the wrong question. The better question is: best for which operating style?

- For most people: ccstatusline (https://github.com/sirmalloc/ccstatusline)
- For plugin-first polish: claude-powerline (https://github.com/Owloops/claude-powerline)
- For Starship-heavy Rust users: cship (https://github.com/stephenleo/cship)
- For observability: claude-hud (https://github.com/jarrodwatts/claude-hud)
- For conservative operators: claudeline (https://github.com/fredrikaverpil/claudeline)
-

- For shell lovers who care about quick visual satisfaction more than feature count: the SippieCup (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871)-style Ghostty lineage is still hard to beat

That last point is worth ending on. The most satisfying statuslines are not always the ones with the largest README or the most widgets. Sometimes the winner is just the one you can understand without really needing to read it.

---

## About This Research

This analysis was prepared as part of an ongoing research workflow focused on agent infrastructure. Supporting evidence and outbound references are versioned alongside the report so the reasoning stays auditable over time.

Research by Yiğit Konur • April 16, 2026 • *methodology*

---

SOURCES


[1] Claude Code statusline docs (https://docs.anthropic.com/en/docs/claude-code/statusline)

[2] anthropics/claude-code statusline issues (https://github.com/anthropics/claude-code/issues?q=is%3Aissue+statusline)

[3] sirmalloc/ccstatusline (https://github.com/sirmalloc/ccstatusline)

[4] Haleclipse/CCometixLine (https://github.com/Haleclipse/CCometixLine)

[5] Owloops/claude-powerline (https://github.com/Owloops/claude-powerline)

[6] jarrodwatts/claude-hud (https://github.com/jarrodwatts/claude-hud)

[7] stephenleo/cship (https://github.com/stephenleo/cship)

[8] fredrikaverpil/claudeline (https://github.com/fredrikaverpil/claudeline)

[9] kamranahmedse/claude-statusline (https://github.com/kamranahmedse/claude-statusline)

[10] daniel3303/ClaudeCodeStatusLine (https://github.com/daniel3303/ClaudeCodeStatusLine)

[11] felipeelias/claude-statusline (https://github.com/felipeelias/claude-statusline)

[12] Astro-Han/claude-pace (https://github.com/Astro-Han/claude-pace)

[13] wolfdenpublishing/pyccsl (https://github.com/wolfdenpublishing/pyccsl)

[14] kcchien/claude-code-statusline (https://github.com/kcchien/claude-code-statusline)

[15] chongdashu/cc-statusline (https://github.com/chongdashu/cc-statusline)

[16] leeguooooo/claude-code-usage-bar (https://github.com/leeguooooo/claude-code-usage-bar)

[17] ryoppippi/ccusage (https://github.com/ryoppippi/ccusage)

[18] SippieCup statusline gist (https://gist.github.com/SippieCup/0cd256789b6350196fc34c6d0ac09871)

[19] jtbr statusline gist (https://gist.github.com/jtbr/4f99671d1cee06b44106456958caba8b)

-