

Kaku

AI コーディング向けに設計した高速ターミナルエミュレータ。初期設定のまま使え、Lua で深く拡張できる、軽量で高性能な実装です。

Kaku とは

Kaku（書く）は、思考をかたちにする行為を指す言葉です。WezTerm をベースに深く最適化し、実用的なデフォルトと完全な Lua カスタマイズ性、軽快な操作感を両立したターミナルです。

4800+

GitHub Stars

Rust

主要言語

40 MB

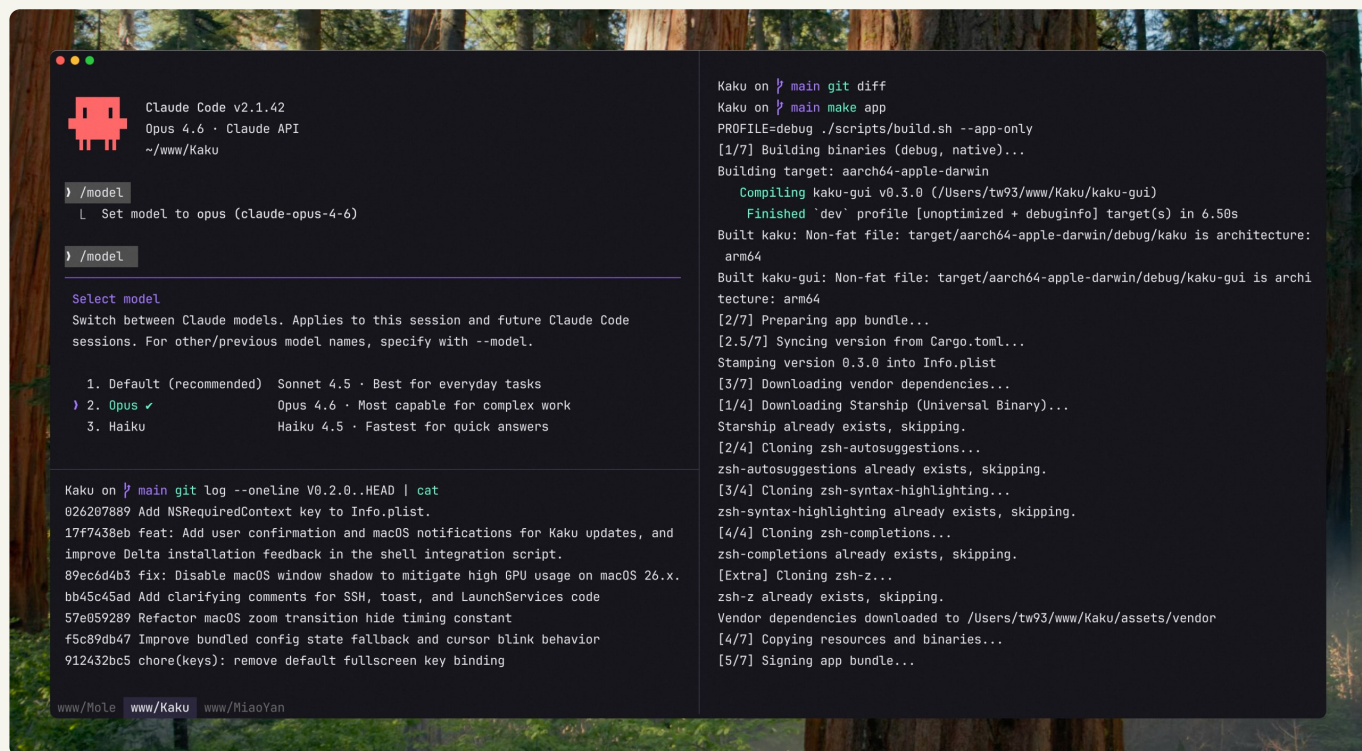
実行ファイル

macOS

重点プラットフォーム

Kaku は三部作の一角です。Kaku（書く）がコードを書き、Waza（技）が習慣を鍛え、Kami（紙）が文書を仕上げます。私は長く CLI 中心で仕事をしてきました。Alacritty を長年使ったあと、AI 補助コーディングへ比重が移るにつれ、より強いタブ管理と内蔵 AI 連携が必要になりました。Kitty、Ghostty、Warp、iTerm2 も試しましたが、性能優先、デフォルトで使いやすい、Lua で全面的に拡張できる、というバランスに最も近かったのが Kaku です。

WezTerm は強力なエンジンです。Kaku ではその上で 2 点に集中しました。不要な重量の削減と、AI ワークフローの直接統合です。設定は WezTerm Lua API と互換で、既存設定を移行せず再利用できます。Starship、Delta、Lazygit、Yazi も最初から連携済みです。macOS では、Traffic Light のタブバー統合、システム外観への追従、ネイティブフォント描画、Apple notarization まで含めて、導入直後から実用状態になります。



主要機能

Kaku は複雑な初期設定なしで使い始められます。同時に WezTerm Lua API 互換を維持し、深いカスタマイズにも対応します。

ゼロ設定で起動

標準フォントは JetBrains Mono。macOS の描画特性に合わせて調整済みです。ディスプレイ解像度を自動判定し、低解像度は 15px、高解像度は 17px、行高 1.28 を自動適用します。

テーマ自動切替

macOS の外観設定に連動して Kaku Dark / Kaku Light を自動切替します。選択色、字重、色上書きテーブルも同時更新され、`color_scheme` で固定運用もできます。

厳選シェルツール

zsh プラグインを自動読み込みします。`z` ディレクトリ移動、`zsh-completions`、構文ハイライト、補完候補、Smart Tab に対応し、fish 連携も可能です。

高速で軽量

シンボル削減と機能整理でバイナリを 40% 圧縮しました (67MB -> 40MB)。リソースは 100MB から 80MB へ。JIT 初期化で Shell 初期化時間も半減します (200ms -> 100ms)。

WezTerm 互換設定

WezTerm の Lua 設定をそのまま利用でき、API 互換を保ちます。移行作業は不要です。`kaku.lua` は defaults を先に読み込み、その後 overrides を適用します。

洗練されたデフォルト

選択コピー、クリック可能なパス、履歴表示、ペイン入力ブロードキャスト、バックグラウンド完了表示、Traffic Light のタブバー統合を標準搭載します。

Lazygit 連携

PATH または Homebrew のパスから lazygit バイナリを自動検出します。未コミット変更がある場合は 1 回だけ通知します。

`Cmd + Shift + G`

Yazi 連携

Shell ラッパー `y` で起動し、終了時に作業ディレクトリを同期します。配色は `yazi/theme.toml` に自動反映されます。

`Cmd + Shift + Y`

Remote Files (SSHFS)

アクティブな SSH ペインから接続先を自動判定し、sshfs でリモートファイルシステムをマウントして Yazi で閲覧できます。

`Cmd + Shift + R`

Kaku AI アシスタント

Kaku には AI アシスタントを内蔵しています。2 つのモードで、ターミナル作業の主要な AI ユースケースをカバーします。 `kaku ai` で設定パネルを開き、外部コーディングツールと Kaku Assistant を設定できます。

エラー自動リカバリ

コマンド失敗時に、失敗コマンド + 終了コード + 作業ディレクトリ + git ブランチを LLM へ送信し、修正案をターミナル内に表示します。 `Cmd + Shift + E` でそのまま適用できます。

未発火条件: `Ctrl+C` 中断、help フラグ、素のパッケージマネージャー実行、git pull 競合、非 shell 前景プロセス。危険コマンド（ `rm -rf` 、 `git reset --hard` ）は貼り付け可能ですが自動実行しません。

自然言語からコマンド

プロンプトで `# <説明>` を入力して Enter すると、Kaku が shell に渡す前に行を取得し、現在のディレクトリと git ブランチを添えて LLM に送信します。生成されたコマンドはプロンプトへ戻り、確認してから実行できます。

```
# list all files modified in the last 7 days
# find and kill the process on port 3000
# compress the src folder excluding node_modules
```

assistant.toml 設定項目		対応 AI コーディングツール	
項目	説明	ツール	説明
enabled	true で有効 / false で無効	claude	Claude Code・Anthropic
api_key	プロバイダー API Key	codex	OpenAI Codex CLI
model	モデル識別子 (例: DeepSeek-V3.2)	gemini	Gemini CLI・Google
base_url	OpenAI 互換 API のベース URL	copilot	GitHub Copilot CLI
custom_headers	プロキシ追加 HTTP ヘッダー	kimi	Kimi Code・Moonshot

性能比較

シンボル削減、機能整理、リソース最適化により、Kaku は機能を維持したままサイズと起動時間を大幅に改善しました。macOS 専用設計でプラットフォーム統合も深めています。

指標	上流 WezTerm	Kaku	最適化手法
実行ファイルサイズ	~67 MB	~40 MB	シンボル削減 + 機能整理
リソース容量	~100 MB	~80 MB	リソース最適化 + 遅延ロード
起動遅延	標準	即時	JIT 初期化
Shell 初期化時間	~200ms	~100ms	環境設定フロー最適化

バイナリ最適化	コンパイル時に未使用の feature flags を削除し、symbol stripping とデバッグ情報削減を実施。Rust release profile は LTO + codegen-units=1 で最適化を最大化します。
遅延ロード戦略	Shell ツール群（Starship、Delta、Yazi テーマ同期）は just-in-time 初期化を採用。最初の Kaku セッションでのみ起動し、システム全体の shell 起動時間を悪化させません。
macOS 統合	Traffic Light のタブバー統合（INTEGRATED_BUTTONS RESIZE）、システム外観連動、Apple notarization（警告なし）、macOS ネイティブフォント描画スタックを提供します。

インストールと使い方

Kaku はシンプルな導入手順と直感的なショートカット体系を提供します。初回起動時に shell 環境も自動設定されます。

1 インストール

GitHub Releases から最新の DMG を取得してアプリケーションフォルダへ移動するか、Homebrew で導入します。

```
brew install tw93/tap/kaku
```

2 初回起動

Kaku は Apple notarization 済みです。セキュリティ警告なしで起動でき、初回起動時に shell 環境を自動設定します。

3 設定カスタマイズ

Cmd + , で設定パネルを開くか、kaku config で Lua 設定ファイルを編集します。

主要ショートカット:

新規タブ	Cmd + T	新規ウィンドウ	Cmd + N
タブ/ペインを閉じる	Cmd + W	タブ切替	Cmd + Shift + [/]
ペイン縦分割	Cmd + D	ペイン横分割	Cmd + Shift + D
ペイン移動	Cmd + Opt + 矢印	Lazygit を開く	Cmd + Shift + G
Yazi を開く	Cmd + Shift + Y	AI パネル	Cmd + Shift + A
AI 提案を適用	Cmd + Shift + E	画面クリア	Cmd + K

CLI コマンド一覧

コマンド	説明
kaku ai	AI 設定パネルを開き、Kaku Assistant と外部コーディングツールを設定
kaku config	既定エディタで ~/.config/kaku/kaku.lua を開く
kaku doctor	診断を実行し、shell 連携、PATH、任意ツールの導入状態を確認
kaku update	最新バージョンの確認と更新を実行
kaku init	shell 連携を設定し、Starship / Delta / Lazygit / Yazi を任意で導入

Kaku はオープンソースです

GitHub github.com/tw93/Kaku

公式サイト kaku.app

インストール `brew install tw93/tap/kaku`

Stars 4800+ 増加中

ドキュメント [完全ドキュメントと FAQ](#)

作者 [@HiTw93](#)

