PRAXIS-LEITFADEN · CLAUDE CODE

Die 5 mächtigsten Claude-Code-Plugins — und wie du sie heute installierst

Alle fünf Plugins mit Copy-Paste-Befehl, die richtige Installations-Reihenfolge — und der eine Fehler, der dir die halbe Wirkung klaut.

Praxis-Leitfaden · Claude Code · Stand August 2026

DU BEKOMMST DREI DINGE

01 Alle fünf Plugins mit **Projektquelle und Befehl** — und dem, was sie wirklich tun.

02 Die **Copy-Paste-Blöcke**, mit denen du jedes davon in Minuten laufen hast.

03 Die **Reihenfolge** — plus den einen Fehler, an dem die meisten scheitern.

KURZ VORWEG, DAMIT DU WEISST, WAS AUF DICH ZUKOMMT

→ **OmniRoute** gibt dir Nutzung, wenn dein Limit leer ist.

→ **claude-mem** gibt Claude ein Gedächtnis.

→ **Headroom** senkt deinen Tokenverbrauch.

→ **Claude Code Setup** sagt dir, was dein Projekt überhaupt braucht.

→ **Task Observer** baut und verbessert deine Skills, während du arbeitest.

Zusammengestellt und gestaltet von Masko Consulting. Die beschriebenen Projekte stehen unter den Lizenzen ihrer jeweiligen Urheber; Angaben zu Sternen und Kennzahlen entsprechen dem Stand August 2026.

TEIL 1 · UNBEGRENZTE NUTZUNG

OmniRoute

Das Projekt `omniroute` ist ein freies AI-Gateway: **ein einziger Endpoint, hinter dem 237 Provider hängen.** Über 50 davon haben einen Free-Tier, 11 sind dauerhaft gratis. Dokumentiert sind laut Projekt rund **1,6 Milliarden kostenlose Tokens pro Monat**.

WAS DAS PRAKTISCH HEISST

Du redest weiter mit deinem gewohnten Tool. Ist dein Kontingent bei einem Provider leer, schaltet OmniRoute automatisch zum nächstbesten Modell weiter — **Auto-Fallback** statt „Limit erreicht, komm in fünf Stunden wieder". Es verbindet nicht nur Claude Code, sondern auch Codex, Cursor, Cline und Copilot über denselben Endpoint.

IN DREI SCHRITTEN EINGERICHTET

SCHRITT 1 · INSTALLIEREN

```
npm install -g omniroute
```

SCHRITT 2 · STARTEN

```
omniroute
```

Schritt 3 — Base-URL eintragen. Dein Tool zeigt ab jetzt auf den lokalen Gateway statt direkt auf einen Anbieter:

SCHRITT 3 · BASE-URL

```
http://localhost:20128/v1
```

GUT ZU WISSEN

Der Kern ist nicht „gratis", der Kern ist **Auto-Fallback**. Ein leeres Kontingent bricht deine Session nicht mehr ab — du arbeitest weiter, nur über ein anderes Modell. Wer das einmal live erlebt hat, macht es nie wieder anders.

TEIL 2 · GEDÄCHTNIS ÜBER SESSIONS

claude-mem

Das Projekt `claude-mem` — rund 90.000 Sterne, Lizenz Apache 2.0 — gibt Claude Code **ein Gedächtnis, das eine Session überlebt**.

WIE ES ARBEITET

1. **Erfassen** — es beobachtet deine Tool-Nutzung während der Arbeit.
2. **Komprimieren** — das Erfasste wird per Claude Agent SDK verdichtet, nicht roh weggeschrieben.
3. **Injizieren** — beim Start einer neuen Session bekommt Claude den relevanten Kontext direkt eingespielt.

Gespeichert wird **lokal**: eine SQLite-Datenbank für die Struktur und eine Chroma-Vektordatenbank mit lokalen Embeddings für die Ähnlichkeitssuche.

INSTALLIEREN

TERMINAL

```
npx claude-mem install
```

Danach Claude Code einmal **neu starten**. Erst nach dem Neustart hängen die Hooks drin.

> **ACHTUNG · NEUSTART NICHT ÜBERSPRINGEN**
>
> Der häufigste „geht nicht"-Fall ist kein Bug: Der Installer ist durchgelaufen, aber die laufende Session kennt das Plugin noch nicht. Beenden, neu starten, dann erst urteilen.

TEIL 3 · WENIGER TOKENS, GLEICHE ANTWORT

Headroom

Das Projekt `headroom` — rund 65.000 Sterne, Lizenz Apache 2.0 — sitzt zwischen dir und dem Modell und **komprimiert alles, was dorthin unterwegs ist**: Tool-Outputs, Logs, Dateien, RAG-Chunks.

WAS ES BRINGT

CODING-AGENTS

Rund **20 % weniger Tokens** im laufenden Betrieb.

JSON-LASTIGES

60 bis 95 % weniger — dort schlägt es am stärksten durch.

Und das Entscheidende: **gleiche Antworten**. Du zahlst weniger für dasselbe Ergebnis, nicht für ein schlechteres.

INSTALLIEREN

PYTHON · AB 3.10

```
pip install "headroom-ai[all]"
```

NODE

```
npm install headroom-ai
```

CLAUDE CODE WRAPPEN

Der einfachste Weg — Headroom startet einen lokalen Proxy und schiebt Claude Code hindurch:

EMPFOHLEN

```
headroom wrap claude
```

Oder du zeigst manuell auf den Proxy:

ALTERNATIVE

```
ANTHROPIC_BASE_URL=http://localhost:8787 claude
```

PRO-TIPP · DER MESSBARE STACK

Headroom und OmniRoute beißen sich nicht — sie greifen an zwei verschiedenen Stellen an. **Headroom senkt, wie viel du verbrauchst. OmniRoute regelt, was passiert, wenn es trotzdem leer ist.** Zusammen ist das der Unterschied zwischen „Limit erreicht" und „läuft einfach weiter".

TEIL 4 · DAS OFFIZIELLE

Claude Code Setup

Zu finden in den offiziellen Claude-Plugins, Ordner `plugins/claude-code-setup`. Das ist kein Community-Bastelwerk, sondern **das offizielle Plugin des Herstellers**.

WAS ES TUT

Es scannt deine Codebase — **package.json**, die Sprachdateien, deine Verzeichnisstruktur — und empfiehlt dir dann pro Kategorie die Top 1 bis 2: MCP-Server, Skills, Hooks, Subagents, Slash-Commands. Statt dreißig Optionen bekommst du **eine kurze, begründete Liste für genau dein Projekt**.

STARTEN

SLASH-COMMAND

```
/claude-code-setup:init
```

Oder ganz in normaler Sprache:

NATÜRLICHE SPRACHE

```
empfiehl Automatisierungen für dieses Projekt
```

GUT ZU WISSEN · READ-ONLY

Das Plugin ändert **keine Dateien**. Es liest, es empfiehlt, fertig. Du kannst es also bedenkenlos auf ein echtes Projekt loslassen — es kann dir nichts kaputt machen. Die Umsetzung bleibt deine Entscheidung.

TEIL 5 · DAS META-PLUGIN

Task Observer

Der Skill `task-observer` stammt aus der Sammlung `one-skill-to-rule-them-all` — Lizenz CC BY 4.0, rund 1.600 Sterne. Er ist das ungewöhnlichste der fünf: ein **Meta-Skill, der alle anderen Skills baut und verbessert** — inklusive sich selbst.

WIE ES ARBEITET

1. **Beobachten** — es schaut deinen Arbeitssessions zu.
2. **Sammeln** — jede Korrektur, die du machst, und jede Urteilsentscheidung landet in einem Observation-Log.
3. **Verbessern** — aus diesem Log werden konkrete Verbesserungen an deinen Skills.

Der Punkt dabei: **Du schreibst deine Skills nicht mehr vorne am Reißbrett.** Sie entstehen hinten, aus dem, was du sowieso schon korrigiert hast.

INSTALLIEREN

Skills leben in Claude Code unter `.claude/skills/<name>/`. Kopier den Skill-Ordner dorthin — projektweit im Repo oder global unter `~/.claude/skills/`:

ZIELPFAD

```
.claude/skills/task-observer/
```

ACHTUNG · REFERENCES/ MITKOPIEREN

Der Skill bringt einen Unterordner **references/** mit. Der muss mit. Wer nur die SKILL.md kopiert, hat einen Skill installiert, dem die halbe Substanz fehlt — und wundert sich, warum nichts Brauchbares rauskommt.

TEIL 6 · REIHENFOLGE UND DER EINE FEHLER

So installierst du sie — und was du dabei falsch machst

Nicht alle fünf an einem Nachmittag. Die Reihenfolge ist nicht egal, weil **zwei davon dein Setup verändern** und drei davon nur dazukommen.

DIE REIHENFOLGE

1. **Claude Code Setup** — zuerst. Read-only, macht nichts kaputt, und du erfährst, was dein Projekt überhaupt braucht, bevor du irgendwas installierst.
2. **claude-mem** — als Nächstes, denn ab hier baut sich dein Gedächtnis auf; je früher, desto mehr sammelt sich an. Neustart nicht vergessen.
3. **Headroom** — wenn dein Verbrauch das Problem ist. Wrappen, ein paar Tage normal arbeiten, dann vergleichen.
4. **OmniRoute** — wenn du regelmäßig ins Limit läufst.
5. **Task Observer** — zum Schluss, denn er lebt von Beobachtungen. Vorher hat er nichts zu beobachten.

ACHTUNG · DER EINE FEHLER

Alle fünf an einem Tag installieren. Danach ist irgendwas komisch, und du hast keine Ahnung, welches der fünf es war — weil zwei zwischen dir und dem Modell sitzen (OmniRoute, Headroom) und zwei in deine Sessions eingreifen (claude-mem, Task Observer). Installier **eins pro Tag** und arbeite einen Tag normal damit. Klingt langsam, ist in Wahrheit der schnellste Weg — weil du keinen halben Tag mit Zurückbauen verbringst.

DEINE ERSTE WOCHE

TAG 1–3

Tag 1 — Claude Code Setup laufen lassen, Empfehlungen lesen, nichts umsetzen.
Tag 2 — claude-mem installieren, neu starten, normal arbeiten.
Tag 3 — prüfen, ob die neue Session dich wiedererkennt.

TAG 4–7

Tag 4 — Headroom wrappen, normal arbeiten.
Tag 5 — OmniRoute, aber nur wenn du wirklich ins Limit läufst.
Tag 6–7 — Task Observer dazu und einfach arbeiten.

Am Ende der Woche schaust du in sein Observation-Log: Da steht schwarz auf weiß, welche Korrekturen du am häufigsten machst. **Genau das sind deine nächsten Skills.**