CLAUDE SKILL

Token-Verbrauch um 58% reduziert

Zehn Tricks die sofort helfen

58%	2x	2 Min	0€
Weniger Token-Verbrauch	Mehr Nachrichten pro Tag	Setup-Zeit	Kosten

Claude liest jedes Mal den kompletten Chat-Verlauf neu. Nachricht 30 kostet 31x so viel wie Nachricht 1. Mit Kontext-Hygiene gewinnst du den größten Teil zurück.

Vorher vs. Nachher

Ohne Token-Hygiene

Limit nach 15-20 Nachrichten · Lange Kontexte die Claude verwirren · Token-Verschwendung durch MCPs und alte Chats

Mit diesen 10 Tricks

58% weniger Verbrauch · Sauberer Kontext = bessere Antworten · Doppelt so viele Nachrichten pro Session

TRICKS 1-4

Kontext-Hygiene

Die wichtigsten Grundlagen für weniger Token-Verbrauch

01 /clear nach jeder Aufgabe

Jede Nachricht in einem langen Chat kostet exponentiell mehr. Nachricht 30 kostet 31x so viel wie Nachricht 1. Neuer Task = neuer Chat. Immer.

02 Plan Mode vor größeren Aufgaben

Wenn Claude in die falsche Richtung coded und du alles verwerfen musst, sind das verbrannte Tokens. Mit /plan analysiert Claude erst die Aufgabe.

03 CLAUDE.md unter 200 Zeilen

Die Datei wird bei jeder Nachricht komplett neu gelesen. Halte sie kurz und nutze sie als Index der auf andere Dateien verweist.

04 MCPs disconnecten

Jeder verbundene MCP-Server frisst tausende Tokens pro Nachricht — selbst wenn du ihn nicht benutzt. Disconnecte alles was du gerade nicht brauchst.

Unsichtbarer Token-Killer
Ein MCP-Server kann bis zu 18.000 Tokens pro Nachricht kosten — selbst wenn du ihn in dieser Session gar nicht benutzt.

TRICKS 5-7

Token-Verbrauch optimieren

Versteckte Kosten sichtbar machen und eliminieren

05 /compact bei 60%

Das fasst deinen bisherigen Chat zusammen und gibt dir wieder Platz. Warte nicht bis Claude es automatisch macht — bei 95% ist der Kontext schon degraded.

06 .claudeignore anlegen

Claude liest dann keine Build-Artefakte, Lock-Files oder node_modules mehr. Erstelle eine .claudeignore im Projekt-Root. Funktioniert wie .gitignore.

07 Thinking-Budget runtersetzen

Claude reserviert standardmäßig 31.999 Tokens fürs Nachdenken. Bei einfachen Aufgaben übertrieben. Setze MAX_THINKING_TOKENS auf 10.000 — spart ~70% an versteckten Kosten. Toggle mit Option+T (Mac) im Chat.

Allein Trick 7 spart ~70% der versteckten Kosten. Die meisten wissen nicht mal, dass Claude standardmäßig 32.000 Tokens nur fürs Nachdenken reserviert — bei jeder einzelnen Nachricht.

TRICKS 8-10

Die letzten Prozente

Fortgeschrittene Optimierungen für Poweruser

08 Sub-Agent Modell auf Sonnet

Wenn dein Hauptmodell Opus ist, laufen Sub-Agents auch auf Opus. Für Suche und Zusammenfassungen reicht Sonnet locker und spart deutlich.

09 5-Minuten-Cache beachten

Claude cached deinen Kontext, aber der Cache läuft nach 5 Minuten ab. Vor jeder Pause /compact — sonst wird alles von Null neu verarbeitet.

10 /context regelmäßig checken

So siehst du wieviel Kontext verbraucht ist und wann du compacten oder clearen solltest.

Alle 10 Tricks im Überblick

#	Trick	Was es spart	Schwierigkeit
01	/clear nach jeder Aufgabe	Token-Explosion stoppen	Einfach
02	Plan Mode nutzen	Verbrannte Tokens	Einfach
03	CLAUDE.md kürzen	Tokens pro Nachricht	Einfach
04	MCPs disconnecten	Bis zu 18.000/Nachricht	Einfach
05	/compact bei 60%	Kontext-Degradierung	Einfach
06	.claudeignore anlegen	Index-Tokens	Mittel
07	Thinking-Budget senken	~70% versteckte Kosten	Mittel
08	Sub-Agents auf Sonnet	Opus-Tokens für Tasks	Mittel
09	Cache-Timing beachten	Doppelte Verarbeitung	Einfach
10	/context checken	Bewusstsein schaffen	Einfach

BEST PRACTICES

Die richtige Reihenfolge

So setzt du alle 10 Tricks zusammen ein

01 Vor jeder Session

Checke mit /context deinen aktuellen Verbrauch. Disconnecte MCPs die du nicht brauchst. Stelle sicher dass dein Thinking-Budget angemessen ist.

02 Bei jeder neuen Aufgabe

/clear → neuer Chat. Kurzer, präziser Prompt. Bei komplexen Tasks erst /plan nutzen bevor Claude loscoded.

03 Während der Arbeit

Bei 60% Kontext: /compact. Vor jeder Pause länger als 5 Minuten: /compact. Regelmäßig /context checken.

04 Einmal einrichten

.claudeignore im Projekt-Root anlegen. CLAUDE.md unter 200 Zeilen halten. In settings.json: MAX_THINKING_TOKENS und SUBAGENT_MODEL konfigurieren.

Die wichtigste Erkenntnis: Wenn du ständig ins Limit läufst, brauchst du nicht immer mehr Claude. Du brauchst besseren Kontext. Diese 10 Tricks geben dir genau das.

SKILL INSTALLIEREN

Caveman in einer Zeile installieren

Kopiere diesen Befehl in dein Terminal

01 **Befehl kopieren und ausführen**

Kopiere den Befehl unten in dein Terminal. Der Skill wird automatisch installiert — kein Setup nötig.

02 **Skill aktivieren**

Danach /caveman in Claude Code eingeben — fertig. Der Skill übernimmt die Token-Optimierung automatisch.

03 **Skill deaktivieren**

Zum Deaktivieren: 'stop caveman' oder 'normal mode'. Jederzeit zwischen aktiviert und deaktiviert wechseln.

```
npx skills add JuliusBrussee/caveman
```

Dein 5-Minuten Aktionsplan

1. settings.json: MAX_THINKING_TOKENS auf 10000, SUBAGENT_MODEL auf sonnet
2. .claudeignore erstellen: node_modules/, dist/, *.lock, .next/
3. CLAUDE.md auf maximal 200 Zeilen kürzen
4. Ungenutzte MCPs disconnecten
5. /clear bei neuem Task · /compact bei 60% · /context checken

Du willst mehr KI-Tools und Strategien die dir einen unfairen Vorteil geben?

Folge @tim_huck_ auf Instagram für tägliche Claude-Tipps, Automatisierungen und KI-Workflows die dein Business auf das nächste Level bringen.

→ **@tim_huck_ auf Instagram** · → **www.shortmedia.agency**