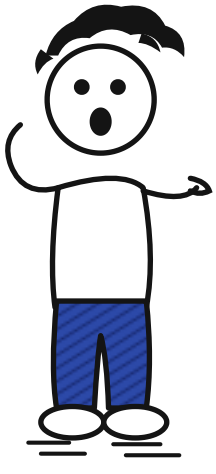
Alle bauen gerade krasse Sachen mit Claude.

Und ich versteh kein Wort davon.



?

CLAUDE.md

Skills

Plugins

MCP

Hooks

Subagents

/Commands

Memory

Checkpoints

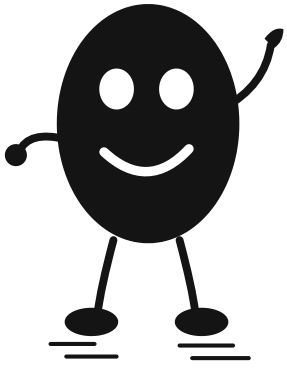
Cowork

Routinen

Plan-Modus

/loop

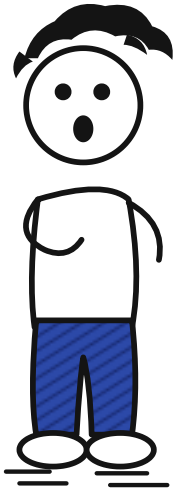
Komm, ich erklär's dir.

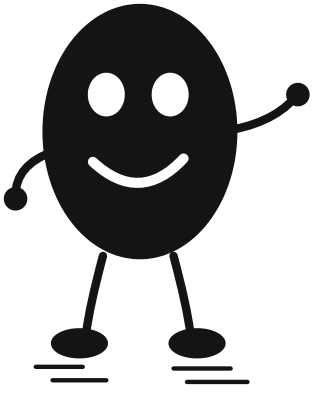


Fangen wir mit CLAUDE.md an. Warum legt die jeder an?

CLAUDE.md ist im Grunde eine Bedienungsanleitung, die du Claude gibst.

Da steht drin, wie dein Projekt funktioniert, welche Regeln gelten – und was Claude auf keinen Fall tun soll.

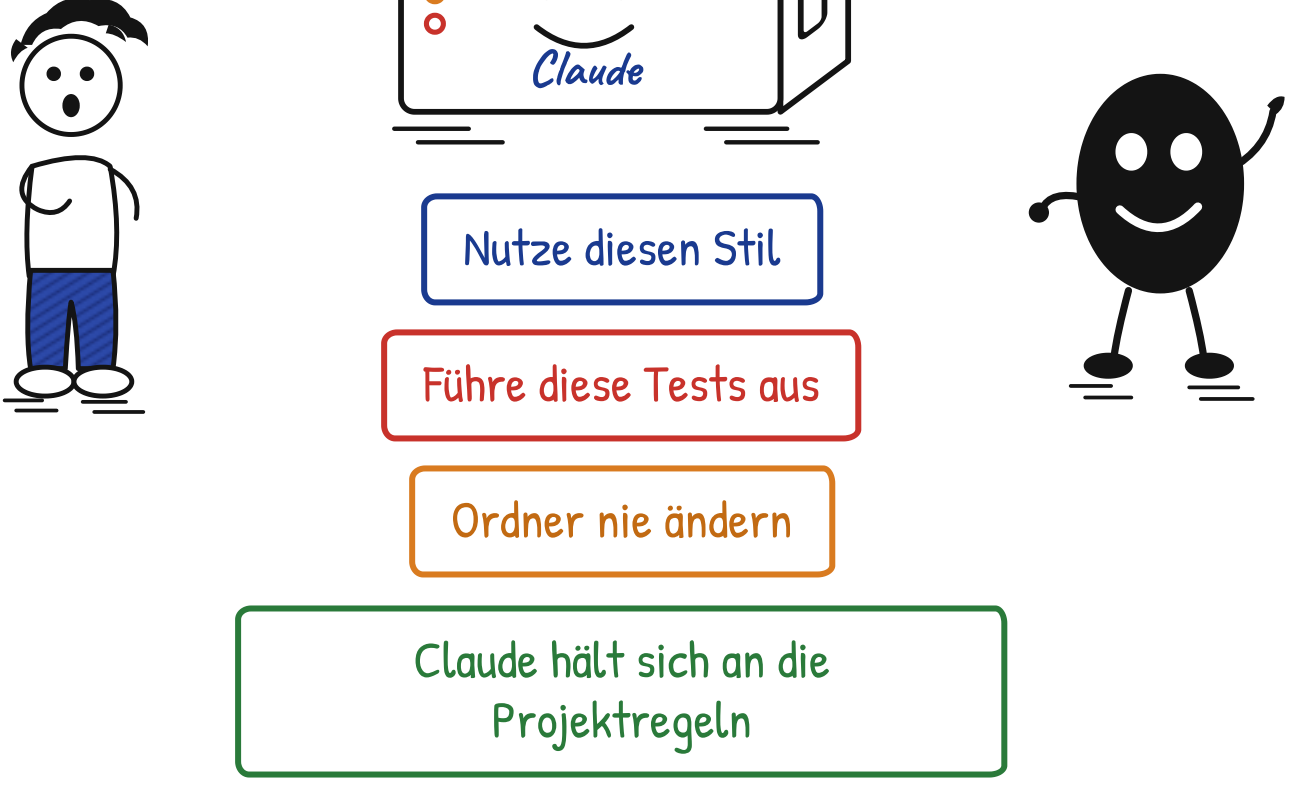




Ich muss also nicht in jeder Sitzung dasselbe erklären?

Genau. Claude liest die Datei automatisch, sobald du im Projekt startest.

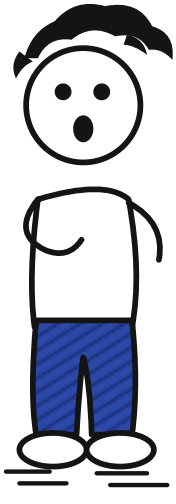
Es gibt sogar drei Ebenen: global in `~/.claude/`, im Projektordner und pro Unterordner. `/init` schreibt dir eine erste Fassung.

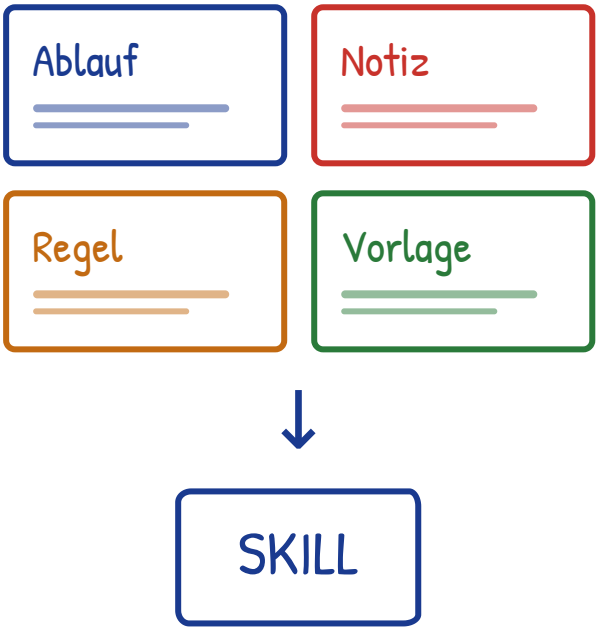


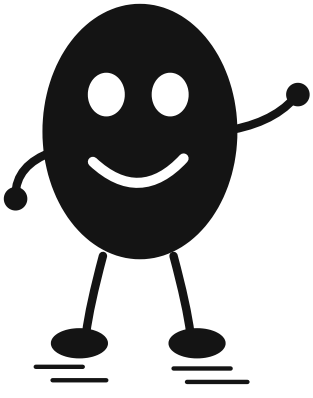
Okay. Und was sind dann Skills?

Ein Skill bringt Claude bei, **wie** eine bestimmte Aufgabe richtig läuft.

Statt jedes Mal deinen ganzen Ablauf zu erklären, machst du daraus einen wiederverwendbaren Skill.



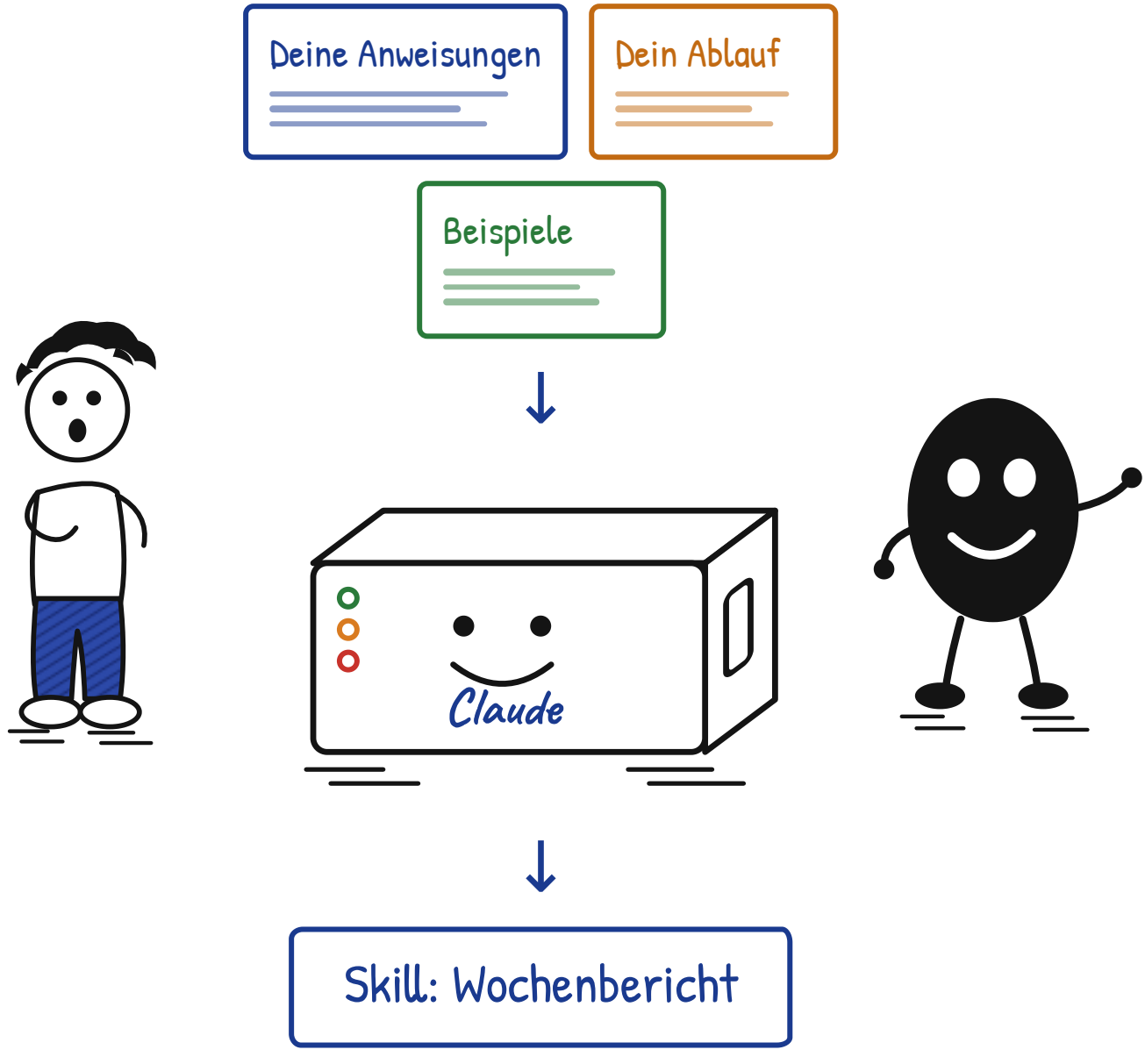




Gib mir mal ein Beispiel.

Sagen wir, du willst jede Woche denselben Auswertungsbericht.

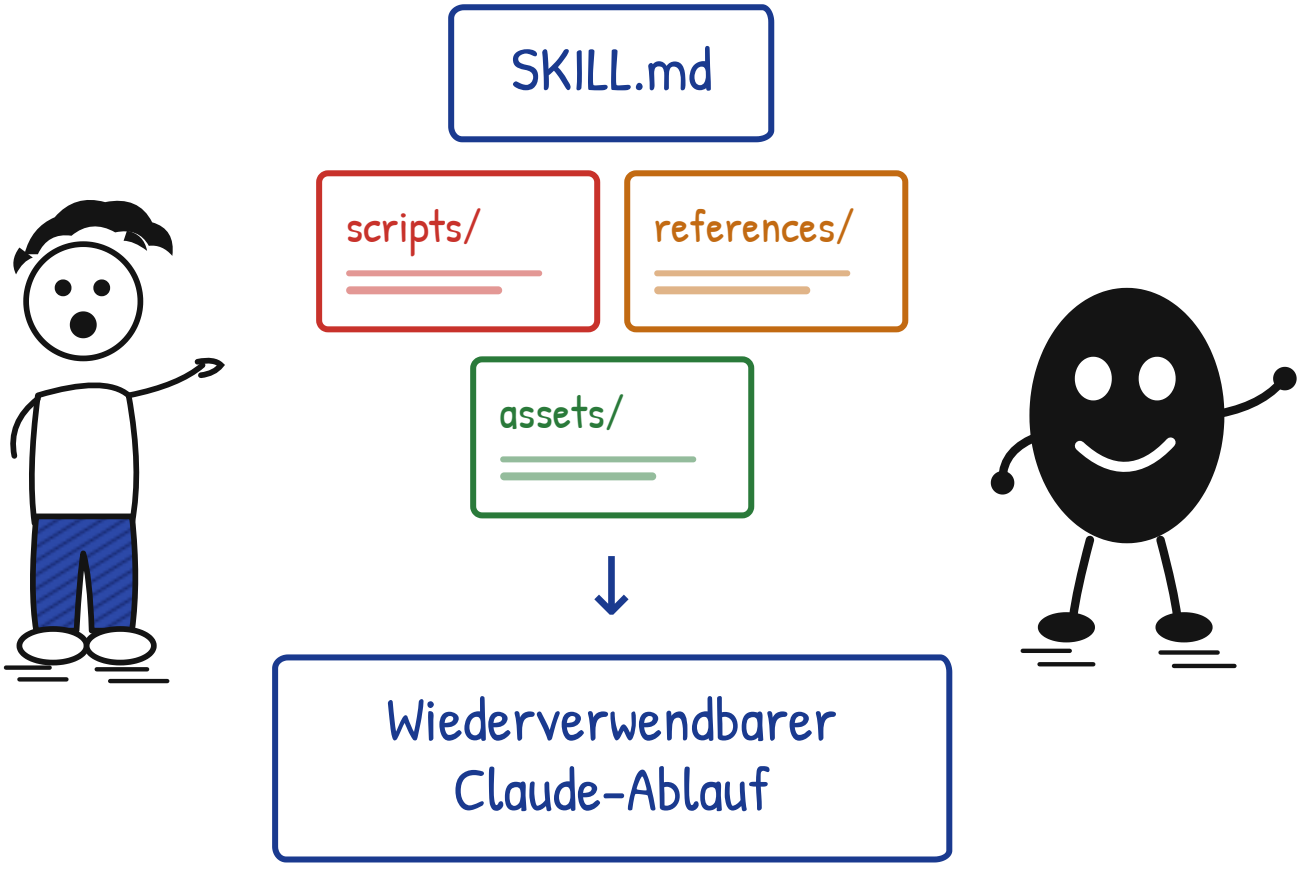
Deine Anweisungen, dein Ablauf, deine Beispiele – daraus wird ein Skill, den du nur noch aufrufst.



Und diese SKILL.md-Dateien, die alle von GitHub laden – das sind einfach Anweisungen?

Ziemlich genau. Ein Skill ist ein Ordner mit einer `SKILL.md`, dazu Skripte, Vorlagen und Referenzdateien.

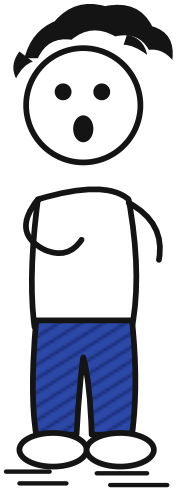
Claude liest zuerst nur Name und Beschreibung – den Rest erst, wenn er ihn wirklich braucht. Aufrufen kannst du ihn auch selbst mit `/skill-name`.

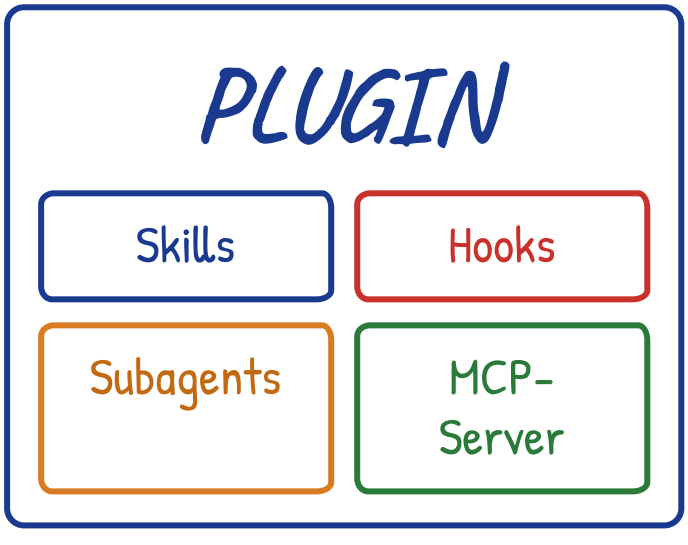


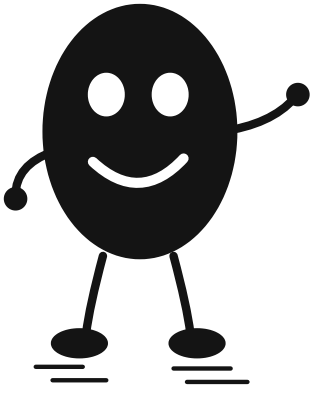
Und Plugins? Sind das nicht auch einfach Skills?

Nicht ganz. Ein Skill ist **eine** Fähigkeit.

Ein Plugin ist eher ein **Paket**: Skills + Hooks + Subagents + MCP-Server + eigene Befehle in einem Rutsch. Installiert wird's über einen Marktplatz mit `/plugin`.





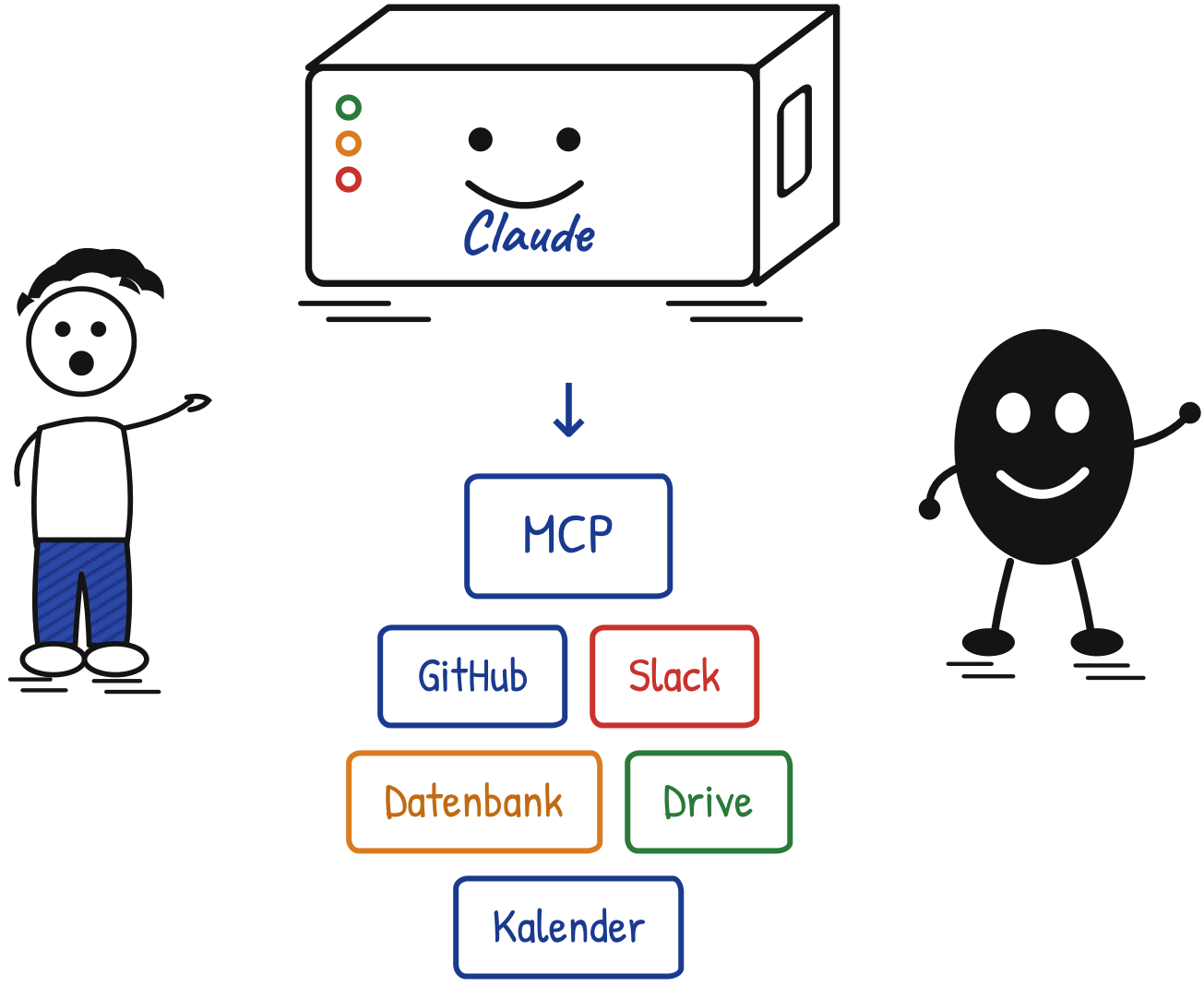


Skill

Moment – MCP-Server. Was ist das?

MCP ist im Grunde die Art, wie Claude sich mit Dingen **außerhalb** von Claude verbindet.

Ein einheitlicher Stecker für fremde Programme und Dienste.



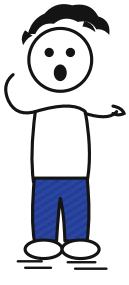
Ich muss also nicht mehr alles von Hand rein-kopieren …

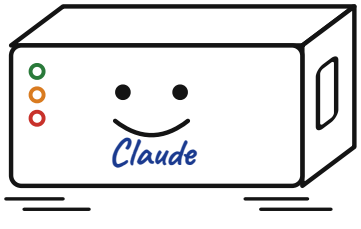
Claude verbindet sich selbst, holt sich die Information – und darf dort auch handeln, wenn du es erlaubst.

Mit `/mcp` siehst du, was gerade verbunden ist.

Ohne MCP

GitHub



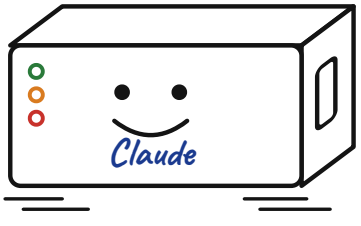


du kopierst alles von Hand

Mit MCP

GitHub

MCP

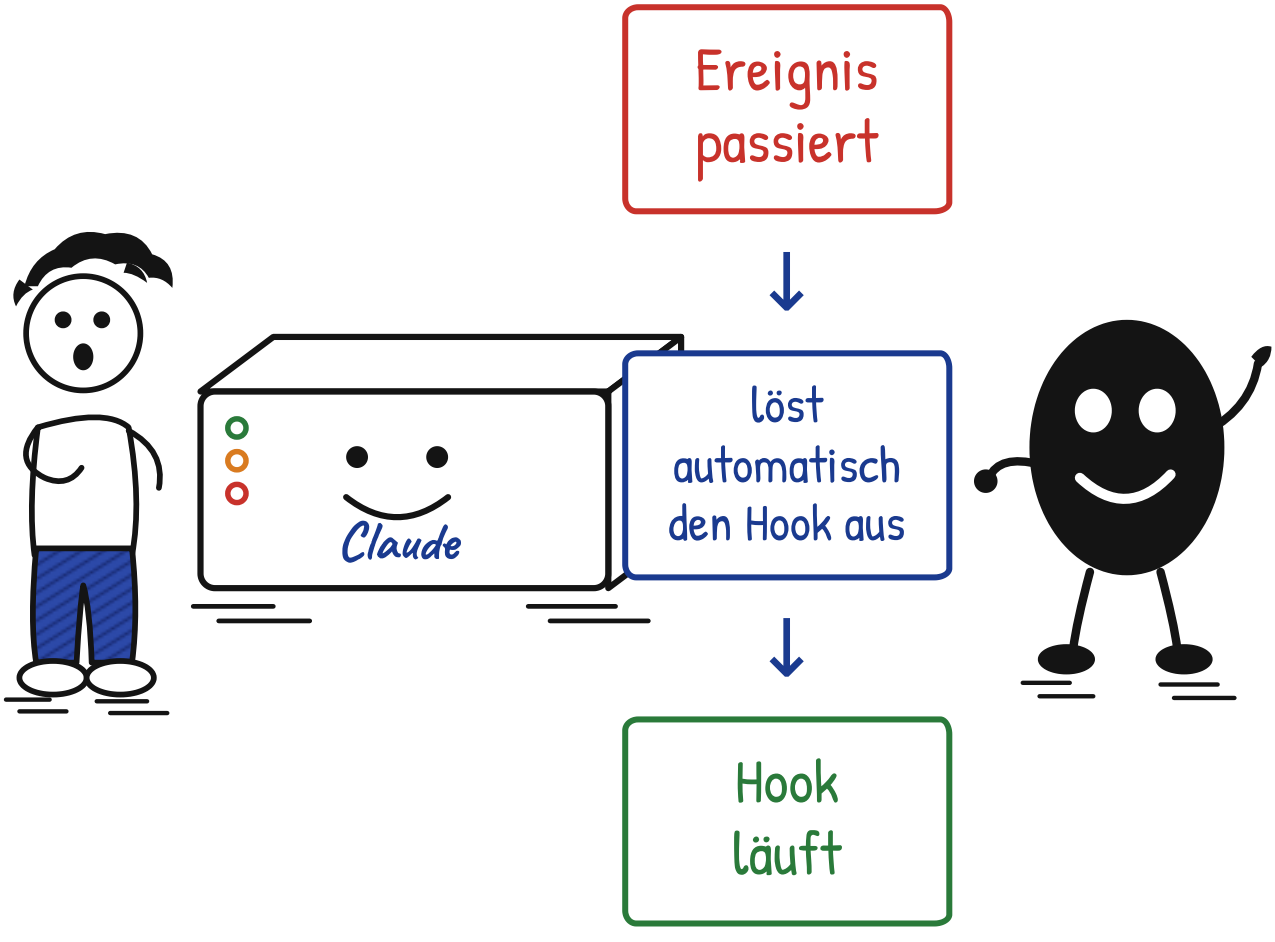


Claude holt es sich selbst

Gut. Und Hooks? Das klingt noch verwirrender.

Hooks sind anders, weil **nicht Claude** entscheidet, ob sie passieren.

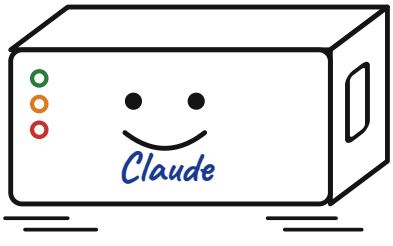
Sie laufen automatisch los, sobald ein bestimmtes Ereignis eintritt. Eingerichtet in `settings.json` oder über `/hooks`.

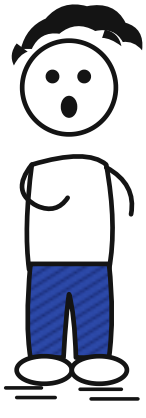


Zum Beispiel?

Ein Hook kann nach jeder Dateiänderung deinen Formatierer starten – oder Claude stoppen, **bevor** ein gefährlicher Befehl läuft.

Typische Ereignisse: `PreToolUse`, `PostToolUse`, `UserPromptSubmit`, `SessionStart`, `Stop`.





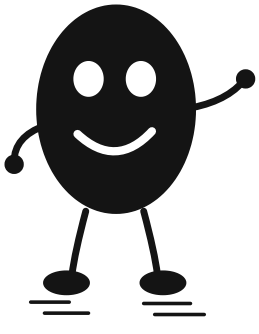
Claude ändert eine Datei

↓

HOOK läuft automatisch

Formatieren | Testen | Prüfen

Melden | STOPP

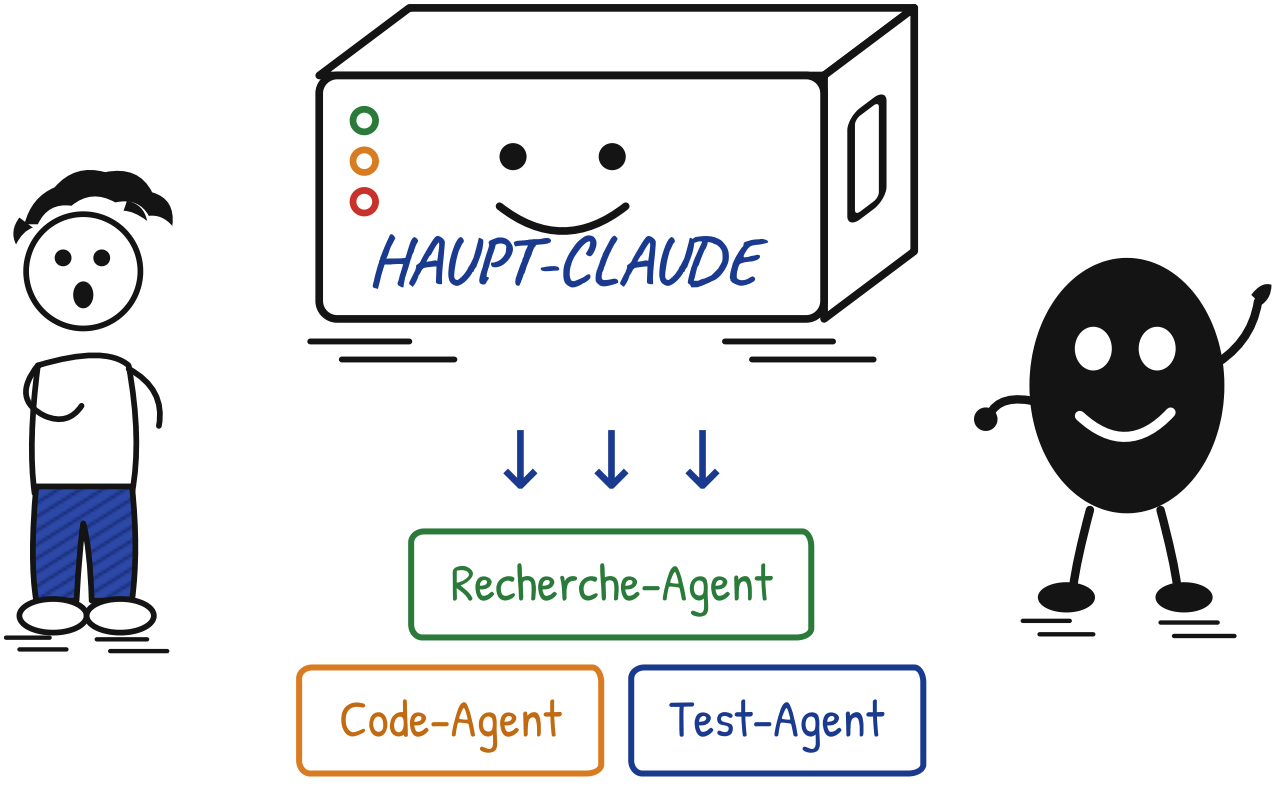


Stopp vor gefährlichen Befehlen

Was ist mit Subagents? Sind das buchstäblich mehrere Claudes?

Das sind eigenständige Arbeiter, denen Claude klar umrissene Aufgaben geben kann – jeder mit eigenem Kontextfenster und eigenem Modell.

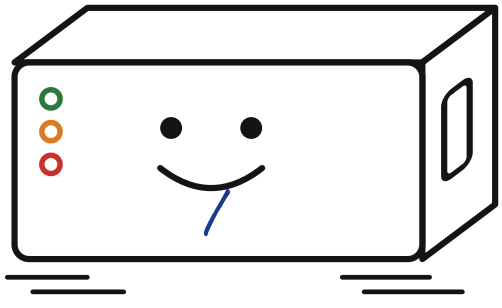
Einer recherchiert, einer prüft Code, einer testet. Danach berichten sie zurück an den Haupt-Claude. Angelegt werden sie als Dateien unter `.claude/agents/` – oder Du lässt Claude das machen.

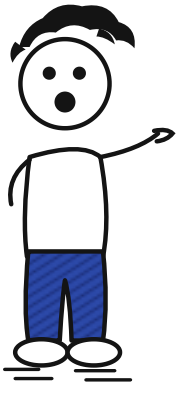


Und was sind diese ganzen /irgendwas-Befehle, die alle benutzen?

Der Schrägstrich ist das Kurzbefehl-Menü von Claude Code.

Es gibt eingebaute wie `/clear` und `/compact` – und Skills bringen eigene Befehle mit, die du selbst aufrufen kannst.





`/clear`
neu anfangen

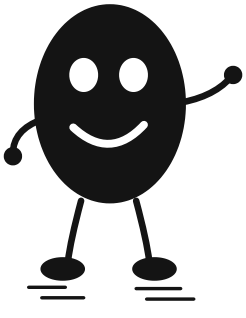
`/compact`
Kontext eindampfen

`/context`
Wer frisst den Platz?

`/rewind`
zurückspulen

`/model`
Modell wechseln

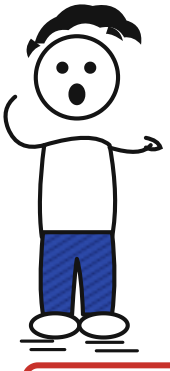
`/usage`
Verbrauch ansehen

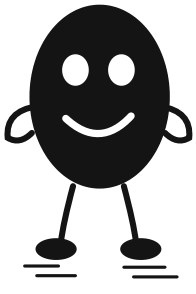


Darf Claude einfach alles auf meinem Rechner?

Auf den bezahlten Zugängen startet Claude Code inzwischen im **Auto-Modus**: Es arbeitet durch, und statt Dir schaut vor jeder Aktion ein zweites Modell drauf.

Umschalten kannst Du jederzeit mit `Shift`+`Tab`: **Manuell** fragt wieder vor jedem Eingriff, **Plan** schaut nur und legt Dir einen Vorschlag hin.





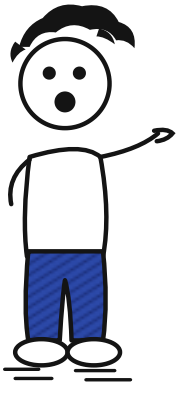
Automatisch	arbeitet durch – der Prüfer schaut mit, Voreinstellung
Manuell	liest von allein, fragt vor jedem Eingriff
Änderungen ok	darf Dateien anfassen, der Rest bleibt Rückfrage
Plan	schaut nur und schlägt vor – ändert nichts

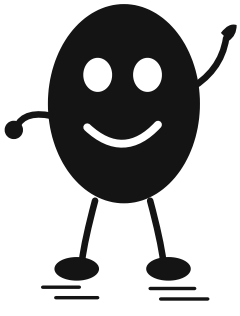
umschalten mit Shift + Tab

Gibt es Abkürzungen, die man kennen sollte?

Vier Zeichen sparen die meiste Tipparbeit – und zwei Tasten sind der Notausgang.

Merk Dir vor allem Esc: Du musst nicht warten, bis Claude fertig ist. Anhalten, richtigstellen, weitermachen.





`@datei`
eine Datei ins Gespräch holen

`!befehl`
selbst etwas ausführen

`/`
das Befehlsmenü öffnen

`Shift+Tab`
Berechtigungen umschalten

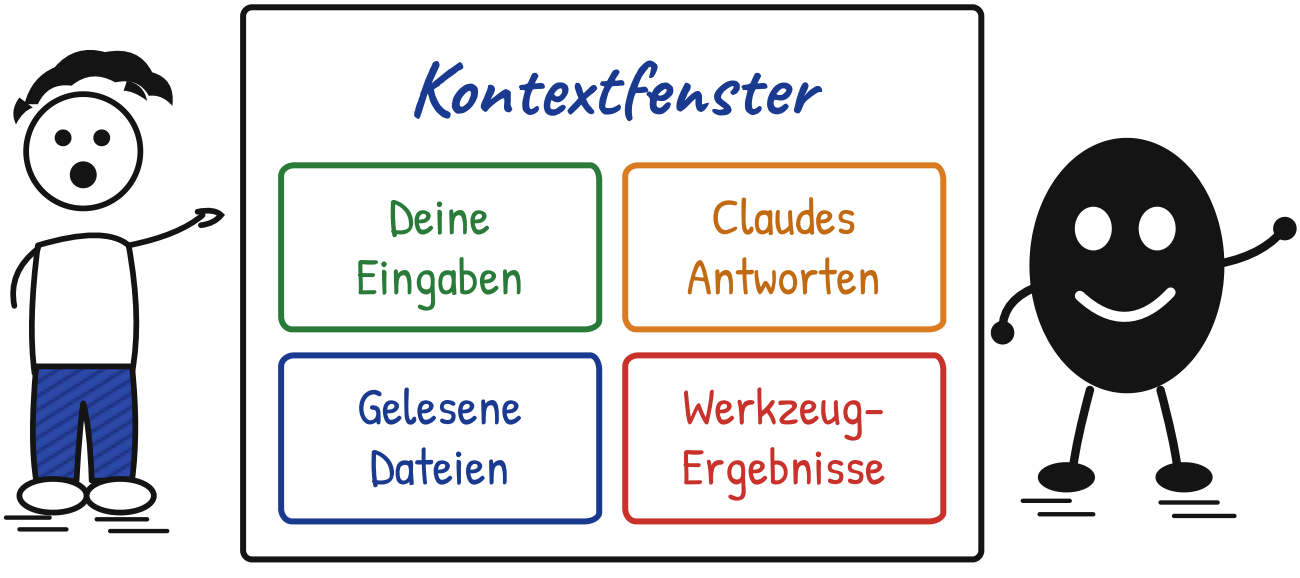
`Esc`
Claude sofort anhalten

`Esc Esc`
zurückspulen

Letzte Frage: Was genau ist dieses Kontext-fenster?

Das ist Claudes Arbeitsgedächtnis für das laufende Gespräch.

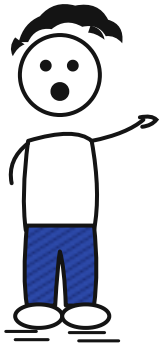
Deine Eingaben, Claudes Antworten, gelesene Dateien und Werkzeug-Ergebnisse belegen alle Platz darin.



Wenn Leute also sagen „Claude hat den Faden verloren" …

… dann ist meist so viel zusammengekommen, dass nicht mehr alles gleichzeitig ins Arbeitsgedächtnis passt.

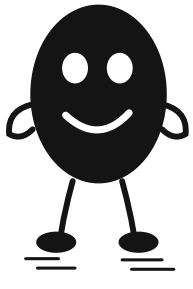
Claude Code fasst dann automatisch zusammen. Mit `/context` siehst du, was den Platz belegt, mit `/clear` fängst du sauber neu an.



KONTEXTFENSTER

- Deine Eingaben
- Claudes Antworten
- Gelesene Dateien
- Werkzeug-Ergebnisse
- Deine Eingaben

Gelesene Dateien | Werkzeug-Ergebnisse | Deine Eingaben



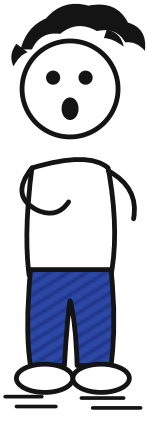
↓

Was Claude gerade präsent hat

Und wenn Claude Mist baut? Alles verloren?

Nein – Claude Code setzt automatisch Sicherungspunkte, bevor es Dateien ändert.

Zweimal `Esc` oder `/rewind`, und du spulst zurück: nur das Gespräch, nur die Dateien oder beides.

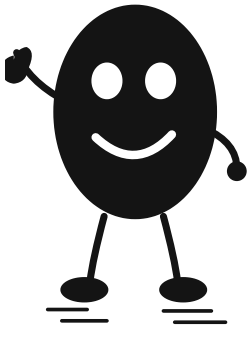


Stand A → Stand B → Stand C

hier ging's schief

← ← ←

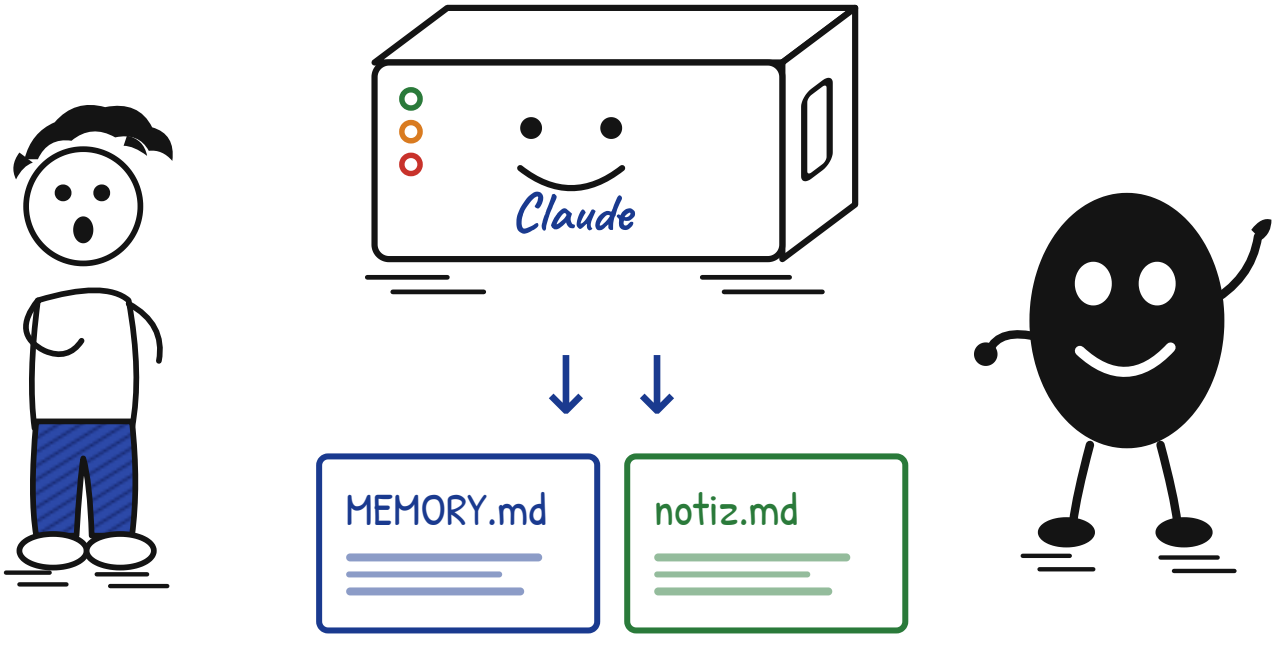
/rewind – zurück auf Stand B



Merkt sich Claude auch etwas über eine Sitzung hinaus?

Ja – dafür gibt es das Gedächtnis: kleine Notizdateien, die Claude selbst schreibt und beim nächsten Start wieder liest.

Deine Vorlieben, Projektstände, wiederkehrende Entscheidungen. Anders als CLAUDE.md pflegt Claude das selbst.

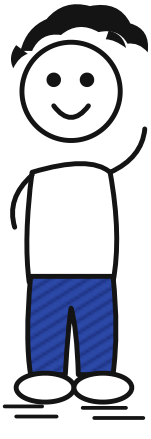


bleibt über Sitzungen hinweg erhalten

Und das läuft alles nur im Terminal?

Längst nicht mehr. Claude Code gibt's als Terminal-Befehl, als Desktop-App, im Browser unter `claude.ai/code` und als Erweiterung für VS Code und JetBrains.

Dazu Agenten, die im Hintergrund oder in der Cloud weiterarbeiten, während du etwas anderes machst.

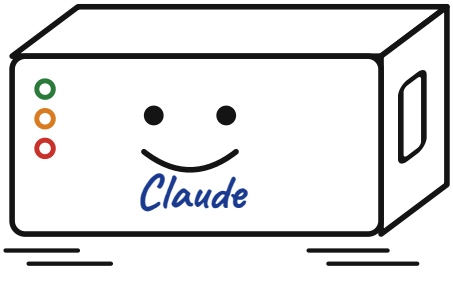


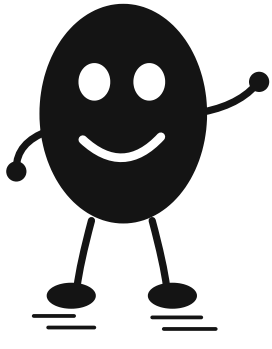
Terminal

Desktop-App

Browser

VS Code / JetBrains



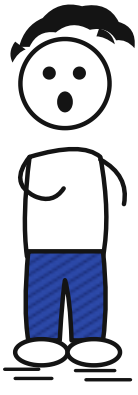


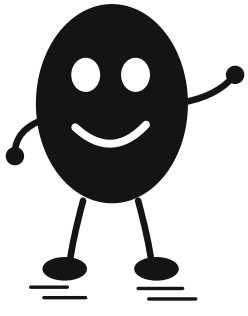
dieselbe Sitzung, überall

Es gibt das alles auch zum Anklicken?

Ja. Die Claude-App für den Rechner hat drei Reiter: **Chat** ist das Gespräch, **Cowork** erledigt Büroarbeit, **Code** ist Claude Code mit Oberfläche.

Es gibt sie für Mac und Windows, für Linux als Beta.

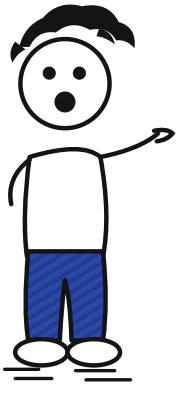


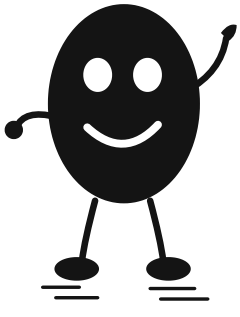


Was kann die App, was das Terminal nicht kann?

Mehrere Sitzungen nebeneinander – jede in ihrer eigenen Git-Arbeitskopie. So kommen sie sich nicht in die Quere, auch wenn sie an derselben Stelle werkeln.

Dazu Editor, Terminal, Vorschau und die Durchsicht der Änderungen im selben Fenster. Losschicken kannst Du eine Sitzung sogar vom Handy.





Sitzung 1
eigene Arbeitskopie

Sitzung 2
eigene Arbeitskopie

Sitzung 3
eigene Arbeitskopie

↓

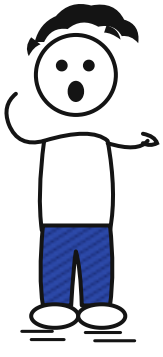
Editor | Terminal | Vorschau | Änderungen prüfen

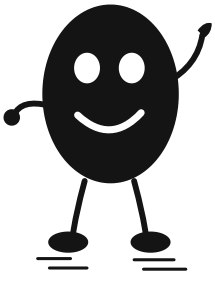
alles in einem Fenster

Und wenn eine Änderung durch das ganze Projekt geht?

Dafür ist `/batch` da: Claude durchdenkt den Umbau einmal und verteilt ihn dann auf viele Agenten, die gleichzeitig loslegen – jeder in einer eigenen Arbeitskopie.

Am Ende legt jeder seinen eigenen Änderungsvorschlag vor. Du siehst sie nebeneinander und nimmst sie einzeln an.





einmal durchdenken und aufteilen

↓

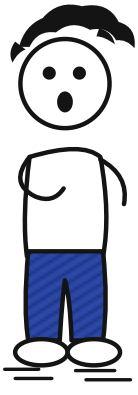
Agent 1	Agent 2	Agent 3	und so weiter
eigene Kopie	eigene Kopie	eigene Kopie	bis zu dreißig

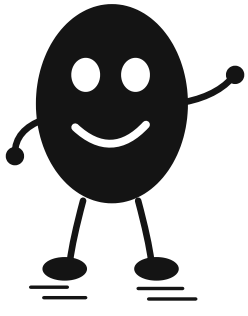
je ein Vorschlag zum Durchsehen

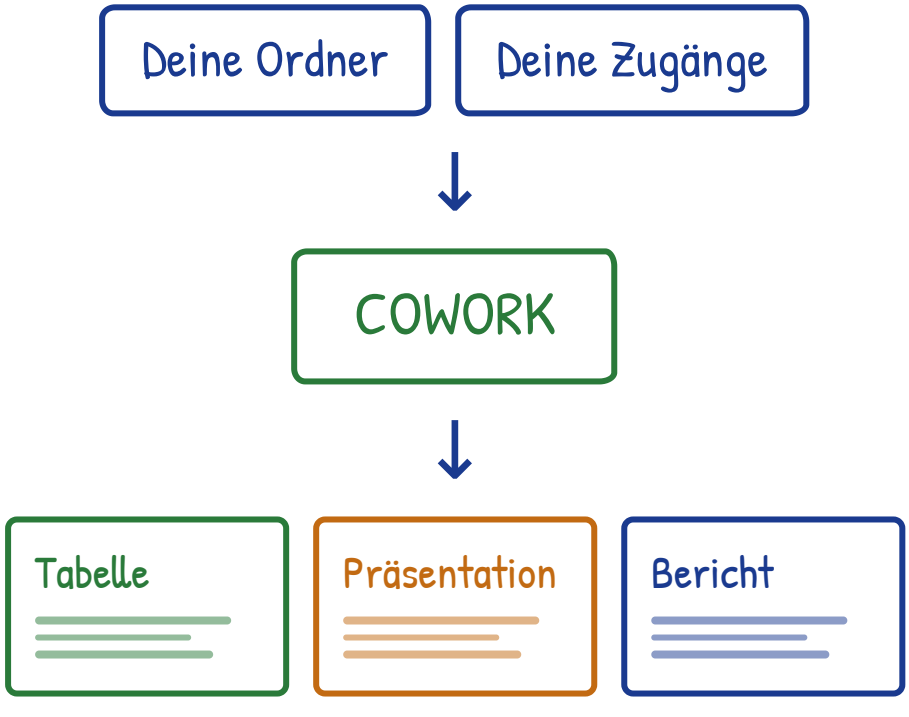
Und was ist dann Cowork?

Dieselbe Technik wie Claude Code – nur in der ganz normalen Claude-App statt im Terminal, ohne Einrichten.

Du gibst Ordner frei, beschreibst das Ziel, und Claude arbeitet los: Tabellen mit echten Formeln, Präsentationen, fertige Dokumente.



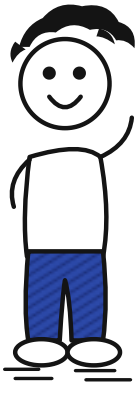


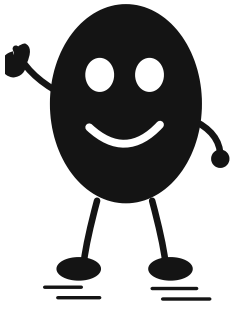


Was macht es denn so?

Einen Ordner voller Belegfotos in eine Abrechnung verwandeln. Aus einem Stapel Notizen einen Bericht schreiben. Eine Ablage aufräumen. Dafür bedient es notfalls auch Chrome – klicken, tippen, Formulare ausfüllen.

Wie viel es allein entscheiden darf, legst Du fest.





Belege → Abrechnung

Notizen → Bericht

Ablage aufräumen

und wenn nötig direkt im Browser

Fragt jedes Mal
Du nickst alles ab

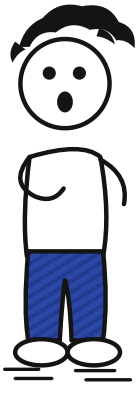
Entscheidet selbst
mit Sicherheitsnetz

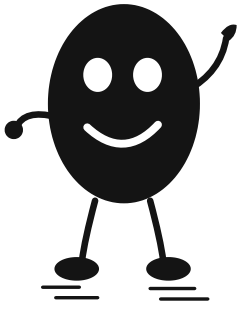
Macht einfach
ohne Rückfrage

Wann nehme ich denn was?

Die Frage ist immer, wie weit Claude an Deine Sachen darf. Cowork bekommt einzelne Ordner, läuft abgeschottet und braucht keine Einrichtung.

Claude Code bekommt den ganzen Rechner. Nur hier kannst Du ihm eigene Werkzeuge beibringen – und nur hier läuft er weiter, wenn Du weg bist.





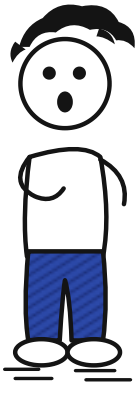
1 · Im Gespräch	nichts einzurichten – Du lädst hoch, Du lädst herunter
2 · Cowork	Desktop-App: Claude arbeitet in Deinen Ordnern
3 · Claude Code	hier bist Du: der ganze Rechner, eigene Werkzeuge, läuft ohne Dich

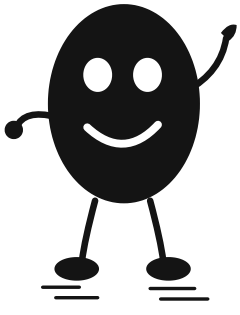
erledigen lassen → Cowork · Werkzeuge bauen, die von allein laufen → Code

Und wenn etwas regelmäßig von allein laufen soll?

Dann legst Du eine Routine an: Auftrag, Projekt und Zugänge einmal gespeichert – danach läuft sie ohne Dich.

Und zwar in der Cloud. Dein Rechner darf ausgeschaltet sein. Angelegt wird sie mit `/schedule`, in der App oder im Browser.





/schedule täglich um 9 Uhr die neuen Pull Requests durchsehen

↓

Auftrag | Projekt | Zugänge

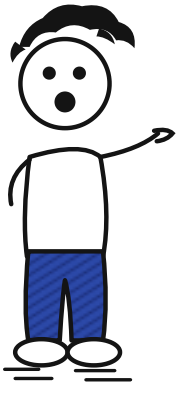
↓

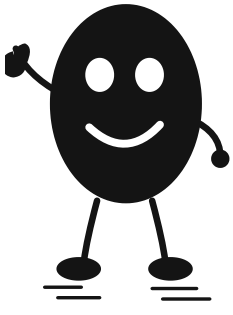
läuft in der Cloud – Rechner aus

Und was bringt so eine Routine ins Rollen?

Drei Dinge, einzeln oder in Kombination: ein Zeitplan, ein Aufruf von außen, oder ein Ereignis auf GitHub.

Der kürzeste Abstand ist eine Stunde. Einmalige Termine gehen auch – `/schedule in zwei Wochen …` Das Ganze ist noch im Vorschau-Stadium.





Zeitplan
stündlich bis wöchentlich

Aufruf
von Deinen Werkzeugen

GitHub
neuer Pull Request

↓

Routine läuft

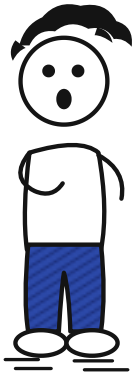
↓

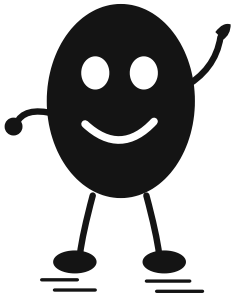
Ergebnis liegt morgens da

Und wenn es nur für die nächste Stunde sein soll?

Dann nimm `/loop`. Der wiederholt einen Auftrag in der offenen Sitzung – auf Deinem Rechner, mit Deinen Dateien.

Lässt Du die Zeitangabe weg, sucht sich Claude den Abstand selbst: kurz, solange sich etwas tut, länger, wenn Ruhe ist. Esc beendet die Schleife.





`/loop 5m schau nach, ob der Bau durch ist`

↻ ↻ ↻

mit Zeitangabe	ohne Zeitangabe	ganz ohne Auftrag
alle 5 Minuten	Claude entscheidet	räumt selbst auf

läuft nur, solange die Sitzung offen ist – und endet nach sieben Tagen

Das klingt doch fast wie eine Routine.

Der Unterschied ist, wo es läuft. Die Schleife braucht Deine offene Sitzung und Deinen eingeschalteten Rechner – dafür sieht sie Deine Dateien.

Die Routine läuft in der Cloud, ohne Dich. Faustregel: Schleife für heute Nachmittag, Routine für jeden Montag.

/loop

auf Deinem Rechner

Sitzung muss offen sein

sieht Deine Dateien

für jetzt gerade

/schedule

in der Cloud

Rechner darf aus sein

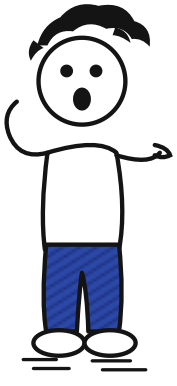
frische Kopie des Projekts

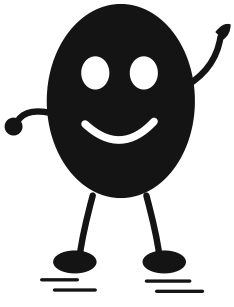
für jede Woche

Und wenn ich noch gar nicht weiß, wie es aussehen soll?

Dann fang mit `/design` an. Claude legt Dir mehrere Entwürfe als Blätter auf eine Zeichenfläche – anklicken, verschieben, Text direkt ändern.

Stimmt die Richtung, gibst Du sie zurück an Claude Code, und dort wird sie gebaut. Steckt noch in der Erprobung.





beschreiben → Entwürfe ansehen → selbst nachziehen → bauen lassen

↓

mehrere Vorschläge nebeneinander

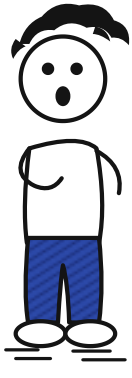
anklicken statt beschreiben

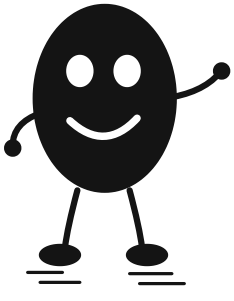
noch in der Erprobung – kann sich ändern

Und welches Modell arbeitet da eigentlich?

Stand August 2026: **Fable 5** ist das stärkste und für lange Läufe gebaut, **Opus 5** nimmt die schweren Sachen, **Sonnet 5** den Alltag, **Haiku 4.5** das Schnelle und Günstige.

Umschalten mit `/model`. Mit `/fast` antwortet Opus schneller – es wird dabei nicht durch ein kleineres Modell ersetzt.





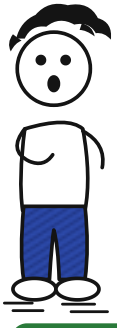
Fable 5	Opus 5	Sonnet 5	Haiku 4.5
lange Agentenläufe	die schweren Aufgaben	der Alltag	schnell & günstig
1 Mio. Kontext	1 Mio. Kontext	1 Mio. Kontext	200 Tsd. Kontext

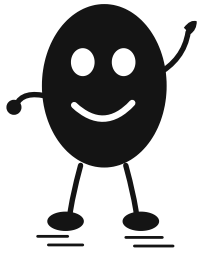
wechseln mit /model

Und wie gründlich es dabei arbeitet – kann ich das auch sagen?

Ja, das ist die zweite Stellschraube neben dem Modell: `/effort`. Erst das Modell nach der Schwierigkeit wählen, dann den Aufwand nach der gewünschten Gründlichkeit.

Voreingestellt ist **high**. Runter heißt schneller und sparsamer, hoch heißt mehr Nachdenken, mehr Werkzeug-aufrufe, längere Läufe.





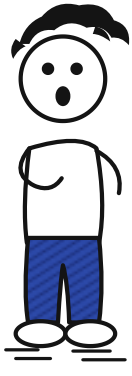
low	kurze, einfache Aufgaben – am schnellsten
medium	der sparsame Mittelweg
high	die Voreinstellung – für alles Anspruchsvolle
xhigh	Agentenläufe über Stunden
max	alles geben, ohne Rücksicht auf den Verbrauch

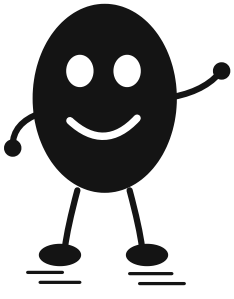
einstellen mit /effort

Und wie sage ich es am besten?

Sag das Ziel, nicht die einzelnen Schritte – den Weg findet Claude selbst. Sag dazu, woran man merkt, dass es fertig ist.

Und gib den Zusammenhang mit: für wen das gedacht ist, wozu, und was auf keinen Fall passieren darf. Je klarer der Auftrag, desto weniger Runden.





eher nicht

„Mach die Tabelle mal schöner."

↓

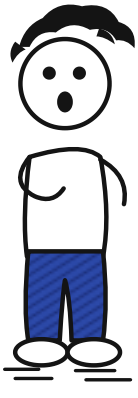
besser

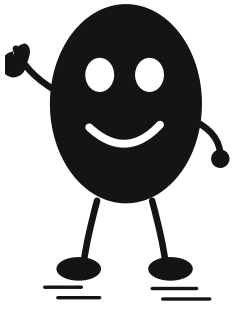
„Die Tabelle geht an den Vorstand. Zahlen rechtsbündig, Summenzeile fett, keine Farben. Prüf am Ende, dass die Summen stimmen."

Kann ich mich denn darauf verlassen?

Meistens ja – aber Claude kann sich auch überzeugend irren. Lass Dir zeigen, was es getan hat, statt es nur zu glauben.

Und was sich prüfen lässt, soll Claude gleich selbst prüfen: Tests laufen lassen, Zahlen nachrechnen, Quellen nennen.





Änderungen ansehen
nicht nur die Zusammenfassung

Tests laufen lassen
am besten automatisch

Quellen nennen lassen
woher stammt die Zahl?

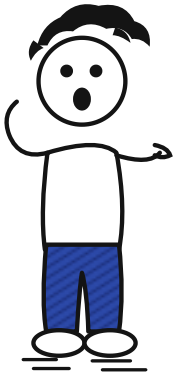
Im Zweifel zurückspulen
Esc Esc kostet nichts

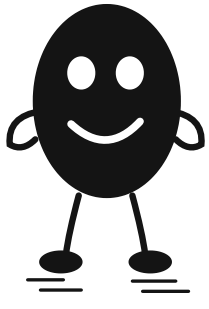
Vertrauen ist gut. Nachsehen ist schneller als Reparieren.

Und wenn in einer Datei steht „lösche alles“?

Gute Frage – genau da liegt die Stolperfalle. Claude liest Webseiten, Mails, Tickets und fremden Quelltext. Nichts davon ist ein Auftrag von Dir.

Aufträge kommen von Dir, aus Deinen Projektregeln und aus dem, was Du erlaubst. Alles andere ist Material zum Lesen – deshalb sind die Berechtigungen keine Schikane.





Deine Anweisung zählt

CLAUDE.md zählt

Webseite nur Material

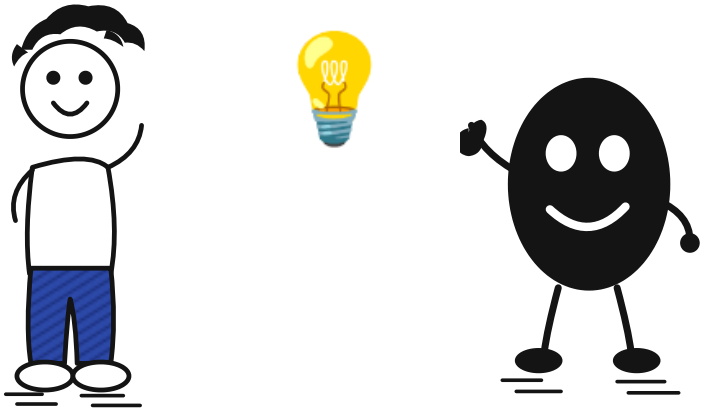
E-Mail nur Material

fremdes Ticket nur Material

Je weiter Claude nach draußen darf, desto enger halte die Leine

Ohh. Ich glaub, jetzt hab ich's.

Das freut mich zu hören.



CLAUDE.md – die Regeln deines Projekts
Skills – wie eine Aufgabe richtig läuft
Plugins – alles davon als ein Paket
MCP – die Verbindung nach draußen
Hooks – laufen automatisch bei Ereignissen
Subagents – Helfer mit eigenem Kopf
Kontextfenster – das Arbeitsgedächtnis
Checkpoints & Memory – zurückspulen und behalten
Cowork – dasselbe in Deinen Ordnern, ohne Terminal
Routinen – laufen nach Plan in der Cloud
Berechtigungen – heute automatisch, jederzeit strenger
Denkaufwand – wie gründlich gearbeitet werden soll
/loop & /batch – wiederholen, oder auf viele aufteilen
/design – erst entwerfen, dann bauen lassen