Claude Code als täglicher Assistent — Setup Guide

Despot Bitschnau
arcneo GmbH, Berlin
Stand: August 2026

Zusammenfassung

Dieser Guide zeigt, wie ich Claude Code im Terminal als persönlichen Alltags-Agenten einsetze. Nicht zum Programmieren, sondern für Mails, Recherche, Notizen, Branding und Organisation. Er führt chronologisch durch das komplette Setup auf dem Mac, von der Terminal-Installation bis zu konkreten Use Cases, die du direkt übernehmen kannst. Wichtig zur Einordnung: Das hier ist bewusst **nicht** mein Setup für Coding-Agenten. Meine Coding-Setups enthalten zwar alles, was hier beschrieben ist, aber darüber hinaus noch deutlich mehr (eigene Projekt-Regeln, Test- und Verifikations-Pipelines, CI-Anbindung und weitere Werkzeuge). Dieser Guide beschreibt ausschließlich das Setup für Nicht-Coding-Aufgaben.

1. Warum Claude Code im Terminal statt Chat im Browser?

Claude Code ist Claude mit Händen. Der Chat auf claude.ai kann dir Texte schreiben, aber Claude Code kann zusätzlich:

- Dateien auf deinem Rechner lesen und schreiben (z. B. deine Obsidian-Notizen)
- Programme ausführen, Webseiten durchsuchen, PDFs bauen
- Über Connectors direkt in Gmail, Google Kalender, Notion, Figma usw. arbeiten
- Sich per CLAUDE.md-Datei dauerhaft merken, wer du bist und wie du arbeitest

Der Effekt: Statt Antworten zu kopieren, erledigt der Agent Dinge. "Fasse meine Mails von heute zusammen und leg die Notiz in Obsidian ab" ist ein einziger Satz.

Und warum nicht Claude Cowork?

Anthropic bietet mit **Claude Cowork** eine Agenten-Variante direkt in der Claude-Desktop-App an, ohne Terminal, ohne Installation. Für den absoluten Einstieg ist das okay, aber für den täglichen Einsatz, wie ich ihn hier beschreibe, ist Claude Code die bessere Wahl:

- **Arbeitet direkt auf deinen echten Dateien.** Cowork läuft in einer abgeschotteten virtuellen Maschine, deine Dateien müssen erst in diese Sandbox hinein und Ergebnisse wieder heraus. Claude Code arbeitet direkt in deinem Dateisystem, also z. B. live in deinem Obsidian-Vault. Genau darauf baut dieses ganze Setup auf.
- **Erledigt Sachen im Hintergrund, ohne dich zu blockieren.** Ein Claude-Code-Auftrag läuft in seinem Terminal-Tab weiter, während du am Rechner ganz normal weiterarbeitest. Claude muss dafür nie deinen Bildschirm oder deine Programme übernehmen, es meldet sich einfach, wenn es fertig ist.
- **Plugins, Skills und eigene Befehle.** Claude Code lässt sich frei erweitern: Plugins installieren, eigene Slash-Commands und Hooks bauen, Berechtigungen fein einstellen. Cowork bringt nur einen festen, vorinstallierten

Funktionsumfang mit.

- **Mehrere Instanzen parallel im Griff.** Mein Kern-Workflow, ein benannter und eingefärbter iTerm2-Tab pro Thema mit jeweils eigener Session, funktioniert nur im Terminal. So laufen problemlos drei, vier Agenten gleichzeitig, ohne dass man durcheinanderkommt.
- **Volle Konfigurierbarkeit.** CLAUDE.md-Dateien pro Ordner, eigene Einstellungen, Sprach- und Voice-Konfiguration, Statuszeile: alles anpassbar.

Kurz: Cowork ist der Mietwagen mit Automatik, Claude Code das eigene Auto. Wer nur einmal schnappen will, was der Agent kann, startet mit Cowork. Wer den hier beschriebenen Alltags-Workflow will, installiert Claude Code, und der Rest dieses Guides zeigt, dass das keine 15 Minuten dauert.

2. Schritt 1: Das richtige Terminal (Mac)

Nimm **iTerm2**, nicht das vorinstallierte Terminal.app. Download: iterm2.com (kostenlos). Einfach die App in den Ordner "Programme" ziehen.

Warum iTerm2:

- **Tabs benennen und einfärben.** Ich habe pro Thema einen Tab offen (z. B. "Mail", "Recherche", "Firma") und gebe jedem eine eigene Farbe. So verlierst du bei mehreren parallelen Claude-Sessions nie den Überblick. Tab benennen: Rechtsklick auf den Tab, "Edit Session Title". Farbe: Rechtsklick, "Edit Tab Color" (oder Cmd+I, dort Titel und Farbe setzen).
- **Native Benachrichtigungen.** Claude Code kann dich über iTerm2 benachrichtigen, wenn eine lange Aufgabe fertig ist und du gerade in einem anderen Fenster bist.
- **Split Panes.** Cmd+D teilt das Fenster, praktisch wenn ein Agent arbeitet und du parallel etwas nachschauen willst.

Typischer Anfängerfehler: Leute bleiben in Terminal.app und wundern sich, warum Benachrichtigungen und Farben nicht funktionieren.

3. Schritt 2: Claude Code installieren

Öffne iTerm2 und füge diese Zeile ein (Enter drücken):

```
curl -fsSL https://claude.ai/install.sh | bash
```

Das installiert Claude Code nativ, ohne dass du vorher Node.js oder Homebrew brauchst. Danach iTerm2 einmal schließen und neu öffnen, dann:

```
claude
```

Wenn "command not found" kommt: Terminal komplett neu starten (nicht nur ein neues Tab). Das ist der häufigste Stolperstein, der Installer trägt den Pfad erst für neue Sitzungen ein.

Anmelden und Team wählen

Beim ersten Start von `claude` öffnet sich der Login im Browser:

1. Mit deinem **Claude-Konto** anmelden (das gleiche wie auf claude.ai). Du brauchst einen Pro-, Max- oder Team-Plan.
2. Achtung, zweiter klassischer Fehler: Beim Login gibt es zwei Wege, "Claude account" (Abo) und "Console account" (API mit Guthaben). Für den Alltag willst du **Claude account**, sonst zahlst du pro Anfrage statt über dein Abo.
3. Wenn du in mehreren Organisationen bist (privat + Firma): Nach dem Login das richtige **Team auswählen**. Falls du im falschen gelandet bist, in Claude `/login` eingeben und neu anmelden, dort lässt sich die Organisation wechseln.

Kurzer Funktionstest: `claude` starten und etwas Einfaches fragen ("Welche Dateien liegen auf meinem Desktop?"). Wenn Claude antwortet und um Erlaubnis für Aktionen fragt, läuft alles.

4. Schritt 3: Grundkonfiguration

In Claude Code tippst du Befehle, die mit `/` beginnen. Die wichtigsten fürs Setup:

/config — die Zentrale

`/config` öffnet die Einstellungen. Dort stellst du ein:

- **Theme:** Dunkles oder helles Farbschema, je nach Terminal-Optik. Einfach durchprobieren.
- **Sprache:** Auf **Deutsch** stellen, dann antwortet Claude immer auf Deutsch, egal wie du fragst. (In der Konfigdatei `~/.claude/settings.json` steht dann `"language": "German"`.)
- **Benachrichtigungen:** Auf iTerm2 stellen, damit du gepingt wirst, wenn eine Aufgabe fertig ist.

Voice — mit Claude sprechen

Das hat meinen Workflow am stärksten verändert: Ich diktiere fast alles, statt zu tippen. In `/config` den **Voice Mode** aktivieren. Ich nutze den "Hold"-Modus: Taste gedrückt halten, sprechen, loslassen, fertig. Diktieren ist gerade bei längeren Aufträgen ("Schreib eine Mail an X, erwähne Y, Ton eher locker...") viel schneller als Tippen. In Kombination mit Sprache = Deutsch fühlt sich das an wie ein Gespräch mit einem Assistenten.

Berechtigungen

Claude fragt anfangs bei jeder Aktion nach Erlaubnis. Das ist am Anfang gut, um ein Gefühl zu bekommen. Später kannst du im Berechtigungsdialog "immer erlauben" wählen oder in `/config` einen entspannteren Modus setzen. Empfehlung: Die erste Woche alles bestätigen, danach lockern.

Der wichtigste Handgriff dabei: **Shift+Tab**. Damit schaltest du in der laufenden Session den Modus um, auf "auto-accept" führt Claude Aktionen direkt aus, ohne jedes Mal nachzufragen. Das mache ich in **jeder Session direkt als Erstes**, sonst sitzt du nur da und drückst Bestätigungen weg. Der Modus gilt immer nur für die aktuelle Session, also gewöhn dir Shift+Tab als ersten Tastendruck nach dem Start an.

5. Schritt 4: Obsidian als Gedächtnis (Second Brain)

Claude Code kann Dateien lesen und schreiben. Obsidian speichert Notizen als einfache Markdown-Dateien in einem Ordner ("Vault"). Diese Kombination ist der Kern meines Setups: **Claude arbeitet direkt in deinem Obsidian-Vault**, ohne Plugin, ohne Schnittstelle, einfach über das Dateisystem.

1. Obsidian installieren: obsidian.md (kostenlos)
2. Einen Vault anlegen, z. B. unter `~/Assistant/vault/`
3. Eine sinnvolle Ordnerstruktur wählen. Meine:

```
vault/
├── Daily/                    Tagesnotizen und Journal
├── Mail/                     Mail-Stilanalyse, Kontakte, Archiv
├── Work/
│   ├── arcneo/               Aktuelle Firma, Unterordner spiegeln das Shared Drive
│   │   ├── 04 HR
│   │   ├── 06 Legal
│   │   ├── 07 Partners
│   │   ├── 08 Marketing & Sales
│   │   ├── 10 Events
│   │   ├── 11 Funding
│   │   └── 14 Agents-Workspace
│   └── Karriere & Zukunft/   Arbeit, aber nicht aktuell
└── Privat/
    ├── Freunde & Familie/
    ├── Wohnung & Verträge/
    ├── Hobbys & Freizeit/    z. B. DJ/, Gaming/
    ├── Bildung & Wissen/
    └── Organisation/         Anlässe, Planungen, Reden
```

Tipp aus der Praxis: Den Work-Ordner genauso benennen wie das Shared Drive der Firma. Dann findet Claude Dinge an beiden Orten unter demselben Pfadnamen.

Wichtig: Obsidian braucht bei mir **keine Community-Plugins**. Alles, was sonst Plugins erledigen (Templates, Auto-Verlinkung, Importe), macht Claude. Obsidian ist nur der schöne Viewer mit Graph-Ansicht und Verlinkung, Claude ist der Autor.

Die CLAUDE.md — dem Agenten eine Identität geben

Im Arbeitsordner (z. B. `~/Assistant/`) legst du eine Datei `CLAUDE.md` an. Die liest Claude bei jedem Start automatisch. Dort steht:

- Wer du bist (Name, E-Mail-Adressen, Firma, Sprachen)
- Wie der Agent antworten soll (kurz, direkt, auf Deutsch)
- Wo was liegt (Vault-Struktur, welche Ordner wofür)
- Regeln ("Nie Mails senden ohne Rückfrage", "Firmen-Drive nur lesen")

Ich habe meinem Assistenten sogar eine Persona gegeben (er heißt bei mir Falco und antwortet mit trockenem Wiener Einschlag). Das ist Geschmackssache, aber es macht die tägliche Arbeit angenehmer. Der wichtige Teil sind die Regeln und Pfade: Je präziser die CLAUDE.md, desto weniger musst du in jeder Session neu erklären.

Startbefehl im Alltag: In iTerm2 `cd ~/Assistant && claude` (oder ein Tab bleibt dauerhaft dort offen).

Advanced: Den Vault als GitHub-Repo sichern

Wenn das Grundsetup läuft, lohnt sich ein Schritt weiter: den Vault-Ordner in ein **privates GitHub-Repository** verwandeln. Das musst du nicht selbst können, das ist selbst schon ein Claude-Auftrag ("Mach aus meinem Vault ein privates GitHub-Repo und richte ein, dass Änderungen automatisch gesichert werden"). Was du davon hast:

- **Backup und Zukunftssicherheit.** Jede Notiz ist versioniert. Nichts geht mehr verloren, auch wenn der Rechner stirbt, und du kannst jede alte Version jeder Notiz wiederherstellen.
- **Mehrere Rechner.** Auf einem zweiten Mac (privat/Arbeit) einfach das Repo klonen, und derselbe Vault mit derselben CLAUDE.md steht dort zur Verfügung. Claude auf beiden Rechnern arbeitet auf demselben Wissensstand.
- **Claude committet selbst.** Bei mir steht in der CLAUDE.md die Regel, dass Claude nach jeder Änderung im Vault automatisch committet und pusht. Das Backup passiert dadurch nebenbei, ohne dass ich je daran denke.

Sensible Ordner (z. B. das private Mail-Archiv) lassen sich per `.gitignore` vom Repo ausnehmen, sie bleiben dann nur lokal.

Advanced: Der Vault als gepflegtes Wiki (drei Schichten)

Der nächste Reifegrad, inspiriert von Andrej Karpathys "LLM Wiki"-Idee: Der Vault ist nicht nur Ablage, sondern ein Wiki, das der Agent aktiv pflegt. Das Setup hat drei Schichten:

1. **Rohquellen (`vault/raw/`).** Artikel, PDFs und Web-Clips landen hier und werden nie verändert. Praktischer Zubringer: die Browser-Erweiterung **Obsidian Web Clipper** macht aus jedem Artikel eine Markdown-Datei. In Obsidian dazu den Attachment-Ordner auf `raw/assets/` stellen und die Funktion "Download attachments for current file" auf einen Hotkey legen (bei mir Ctrl+Shift+D), dann liegen auch alle Bilder lokal und Claude kann sie ansehen.

2. **Das Wiki (der restliche Vault).** Die von Claude geschriebenen, untereinander verlinkten Notizen. Die Regel, die den Unterschied macht: Beim Einlesen einer neuen Quelle legt Claude nicht nur eine Zusammenfassung ab, sondern **aktualisiert auch die bestehenden Seiten**. Es setzt Querverweise, markiert Widersprüche zu älterem Wissen und legt für wiederkehrende Themen eigene Seiten an. So wird Wissen einmal verdichtet und bleibt aktuell, statt bei jeder Frage neu zusammengesucht zu werden (der Unterschied zu klassischem RAG bzw. NotebookLM-artigen Tools).
3. **Das Schema (die CLAUDE.md).** Dort stehen genau diese Pflege-Regeln, damit jede Session sie automatisch befolgt. Du schreibst das Wiki nie selbst: Du lieferst Quellen und stellst Fragen, Claude macht die Buchhaltung. Obsidian ist die Leseansicht (die Graph-Ansicht zeigt dir die Struktur), Claude ist der Autor.

Dazu zwei kleine Helfer, die sich Claude auf Zuruf selbst baut:

- **`vault/log.md`**: eine append-only Chronik (was wurde wann eingelesen, recherchiert, geprüft). Gibt jeder neuen Session sofort Kontext über den Stand des Wikis.
- **`/lint-vault`**: ein eigener Befehl, der das Wiki periodisch auf verwaiste Seiten, tote Links, Widersprüche und fehlende Verknüpfungen prüft und Fixes vorschlägt.

Der Effekt: Gute Antworten verschwinden nicht im Chatverlauf, sondern werden als verlinkte Notiz ins Wiki zurückgeschrieben. Das Wissen kumuliert, jede Recherche macht die nächste besser. (Karpathys Original-Idee: gist.github.com/karpathy/442a6bf555914893e9891c11519de94f)

6. Schritt 5: Connectors (MCP) — Claude an deine Tools anschließen

Connectors verbinden Claude mit externen Diensten. Die meisten richtest du einmal auf **claude.ai** ein (Einstellungen, Connectors), danach stehen sie automatisch auch in Claude Code im Terminal zur Verfügung. Mit `/mcp` siehst du in Claude Code, was verbunden ist.

Meine Connectors und wofür ich sie nutze:

Connector	Wofür
Gmail	Mails suchen, Threads zusammenfassen, Entwürfe im eigenen Stil anlegen
Google Kalender	Termine suchen, anlegen, Zeitvorschläge ("Wann habe ich Freitag Zeit?")
Google Drive	Dokumente suchen und lesen (bei mir: Firmen-Drive nur lesend)
Notion	Seiten lesen, anlegen, Datenbanken abfragen
Figma	Designs lesen UND erstellen (meine LinkedIn-Posts und Pitch-Decks entstehen so)
HubSpot	CRM-Abfragen, Kontakte, Kampagnen
Brave Search	Websuche direkt aus dem Terminal
Context7	Aktuelle Dokumentation von Tools und Bibliotheken
Namecheap	DNS-Einträge meiner Domain verwalten (lokal eingerichtet)

Praxis-Tipp: Nur verbinden, was du wirklich nutzt. Jeder Connector ist ein Zugriff, den der Agent hat. Bei sensiblen Systemen (z. B. Firmen-Drive) in der CLAUDE.md eine Nur-Lesen-Regel festhalten.

7. Schritt 6: Plugins und Skills

Plugins erweitern Claude Code um fertige Fähigkeiten ("Skills"). Installation in Claude Code über `/plugin`, dort den Marketplace durchsuchen. Was bei mir läuft:

- **Superpowers** (offiziell) — **das absolute Must-have, als Erstes installieren.** Superpowers verändert, *wie* Claude arbeitet, um Klassen: Statt draufloszulegen klärt Claude erst die Anforderungen, plant, setzt dann um und verifiziert am Ende sein eigenes Ergebnis. Der Unterschied ist bei jeder nicht-trivialen Aufgabe spürbar, weniger Nachbessern, weniger "das habe ich so nicht gemeint". Wenn du nur ein einziges Plugin installierst, dann dieses.
- **Firecrawl** (CLI + Skills) — **die zweite klare Empfehlung, speziell für Research.** Mein Web-Werkzeugkasten: Suchen, Webseiten sauber auslesen, ganze Doku-Seiten herunterladen, Preisseiten von Wettbewerbern beobachten. Das Highlight sind die Deep-Research-Skills, die liefern Berichte mit echten Quellenangaben statt Halbwissen aus dem Modellgedächtnis. Für alle Recherche-Use-Cases weiter unten ist Firecrawl die Grundlage. Installation: auf firecrawl.dev registrieren, dann die CLI mit den Skills installieren; danach kann Claude jederzeit "das Web lesen".
- **Frontend-Design** (offiziell): Sorgt für hochwertige, nicht nach Vorlage aussehende Gestaltung, wenn Claude Dokumente, Seiten oder Grafiken baut.
- **arcneo-branding** (eigenes Plugin): Meine Marken-Skills, die LinkedIn-Cards, OnePager und Pitch-Deck-Folien direkt in Figma im Firmen-CI erzeugen. Der Punkt dahinter: Man kann sich **eigene Plugins** für die eigenen wiederkehrenden Aufgaben bauen und im Team teilen. Genau das lohnt sich für Branding-Aufgaben enorm.
- **Game-Sounds** (Spielerei): Soundeffekte bei bestimmten Ereignissen. Muss nicht sein, macht aber Laune.

Dazu kommen **Slash-Commands**: eigene Kurzbefehle wie `/note` (Tagesnotiz in den Vault schreiben) oder `/dns` (DNS-Änderung mit Vorher/Nachher-Prüfung). Das sind einfache Textdateien im Ordner `.claude/commands/`, jeder kann sich seine eigenen bauen, einfach Claude darum bitten.

8. Mein täglicher Workflow

So sieht ein normaler Tag bei mir aus:

1. **iTerm2 mit benannten, gefärbten Tabs.** Ein Tab pro Kontext, z. B. blau = "Falco" (persönlicher Assistent), grün = "arcneo" (Firma), orange = "Recherche". Jeder Tab ist eine eigene Claude-Session mit eigenem Gesprächsverlauf.
2. **Voice statt Tippen.** Taste halten, Auftrag diktieren, loslassen. Gerade unterwegs zwischen Meetings der schnellste Weg.
3. **Alles Wichtige landet im Vault.** Ergebnisse (Recherchen, Briefings, Analysen) schreibt Claude als Markdown-Notiz mit Verlinkungen in den passenden Obsidian-Ordner. Abends in Obsidian durchblättern, alles ist da und verlinkt.

4. **Lange Aufgaben laufen im Hintergrund.** Claude arbeitet weiter, iTerm2 meldet sich, wenn es fertig ist. Währenddessen im nächsten Tab das nächste Thema.
5. **Kontext-Fenster sauber halten, immer.** Das ist wichtiger, als es klingt: Claude hat pro Session ein begrenztes "Gedächtnis" (das Kontext-Fenster). Je länger eine Session läuft und je mehr Themen sich darin vermischen, desto langsamer, teurer und unkonzentrierter wird der Agent. Deshalb die eiserne Regel: **Neues Thema = neuer Tab oder `/clear` im bestehenden Tab.** `/clear` leert den Gesprächsverlauf und startet frisch (die CLAUDE.md und alle Einstellungen bleiben natürlich erhalten). Niemals ein neues Thema in eine volle Session hineinquetschen.
6. **Bei sehr anspruchsvollen Aufgaben: ultrathink.** Ein magisches Keyword: Schreibst du das Wort "ultrathink" irgendwo in deinen Auftrag, schaltet Claude in den tiefsten Denkmodus. Claude bekommt dann das maximale Budget an "Nachdenk-Zeit" und überlegt ausführlich, bevor es antwortet oder loslegt: Es wägt Varianten ab, prüft die eigene Logik und plant mehrere Schritte voraus. Das Wort wird im Terminal sogar bunt eingefärbt, damit du siehst, dass es erkannt wurde. Für Alltagsaufgaben unnötig (dauert länger), aber bei komplexen Sachen, etwa einer heiklen Verhandlungs-Mail, einem strategischen Plan oder einer kniffligen Analyse, macht es einen spürbaren Qualitätsunterschied. Beispiel: "ultrathink: Entwirf die Antwort an den Investor, hier ist der Verlauf ..."
7. **Der Agent verbessert sich selbst.** Wenn etwas schiefläuft oder ich eine Vorliebe korrigiere, lasse ich Claude die CLAUDE.md ergänzen. Jeder Fehler passiert nur einmal.

9. Use Cases aus der Praxis (zum Nachbauen)

Der persönliche E-Mail-Stil-Guide

Mein Lieblingsbeispiel. Ich habe mein komplettes Mail-Archiv (rund 2.500 Mails aus drei Jahren, per Google-Takeout-Export) an Claude gegeben. Claude hat sie komplett gelesen und daraus eine **Stil-Bibel** destilliert: Wie ich Leute anspreche (wann "Hallo", wann "Sehr geehrte"), wie ich grüße, Satzbau, Lieblingswörter, was ich nie tue (Emojis, Floskeln), sogar Unterschiede zwischen deutschen und englischen Mails. Ergebnis ist eine Notiz `Mail/Stil.md` im Vault, plus eine Kontaktliste und Few-Shot-Beispiele.

Seitdem gilt: Wenn Claude eine Mail in meinem Namen entwirft, liest er erst den Stil-Guide. Die Entwürfe klingen nach mir, nicht nach KI. Aufwand einmalig ein Nachmittag, Nutzen täglich.

Hobby-Wissensbasis: DJ-Research

Für das DJ-Hobby hat Claude eine komplette Wissensbasis im Vault aufgebaut: Genre- und Szene-Überblick, ein BPM-Index, das Camelot-System fürs harmonische Mixen, Stil-Analysen einzelner Artists (mit eigenem Dossier pro Artist), eine "100 Essential Tracks"-Liste und die Vorbereitung eines Festival-Lineups. Alles recherchiert, strukturiert und untereinander verlinkt. Das Muster funktioniert für jedes Hobby: Kochen, Sport, Sammeln, was auch immer.

Branding und Content direkt in Figma

Über den Figma-Connector plus mein Branding-Plugin erzeugt Claude fertige LinkedIn-Posts, Karussells, One-Pager und Pitch-Deck-Folien direkt in Figma, im richtigen CI (Farben, Schriften, Logo-Platzierung). Aus "Mach einen Ankündigungspost für die Messe" wird ein fertiges Design zum Feinschliff. Auch der LinkedIn-Textstil ist analog zum Mail-Stil als Guide im Vault hinterlegt.

Recherche-Dossiers für andere

Claude recherchiert Studienlagen und Fachthemen und legt sie als saubere Dossiers im Vault ab, bei mir z. B. eine zahnmedizinische Studienübersicht für die Familie oder ein Vergleich von Raketen-Antriebskonzepten aus reinem Interesse. Mit den Firecrawl-Deep-Research-Skills kommen dabei Berichte mit echten Quellenangaben heraus, keine Halluzinationen aus dem Bauch.

Verwaltung und Kleinkram

- **DNS-Verwaltung:** Domain-Einträge per Zuruf ändern, mit eingebauter Vorher/Nachher-Kontrolle.
- **Event-Radar:** Claude beobachtet Branchen-Events und pflegt eine Liste im Vault.
- **Meeting-Nachbereitung:** Notizen diktieren, Claude strukturiert sie und legt sie im richtigen Firmenordner ab.

10. Die häufigsten Fehler von Einsteigern

1. **Terminal nach der Installation nicht neu gestartet** → "command not found". Einfach iTerm2 komplett beenden und neu öffnen.
2. **Mit dem Console-/API-Konto statt dem Claude-Abo angemeldet** → unerwartete Kosten pro Anfrage. Beim Login "Claude account" wählen; korrigieren mit `/login`.
3. **Im falschen Team/der falschen Organisation gelandet** → mit `/login` neu anmelden und die richtige Organisation wählen.
4. **Ohne CLAUDE.md arbeiten** → jede Session bei null anfangen. Die halbe Stunde für eine gute CLAUDE.md ist die beste Investition im ganzen Setup.
5. **Claude im Home-Verzeichnis starten statt im Arbeitsordner** → Claude sieht seinen Kontext nicht. Immer erst `cd ~/Assistant` (oder wo dein Setup liegt), dann `claude`.
6. **Alles in einer einzigen Session machen** → der Verlauf wird lang und unübersichtlich. Pro Thema ein Tab, bei Themenwechsel `/clear` oder neue Session.
7. **Jeden Connector verbinden, "weil man kann"** → unnötige Zugriffe. Nur anschließen, was gebraucht wird, und Schreibrechte bewusst vergeben.

Bei Fragen einfach melden. Und der beste Tipp zum Schluss: Frag Claude selbst. "Wie richte ich X ein?" funktioniert ab dem Moment, in dem die Grundinstallation steht. Am besten gibst du Claude Code direkt nach der Terminal-Installation dieses Dokument und lässt es beim restlichen Setup unterstützen, idealerweise setzt es alles selbst auf: Ordnerstruktur, CLAUDE.md, Vault, Commands. Du bestätigst nur noch.