BREAKDOWN · MAI 2026

Claude · Der komplette Breakdown

Modelle, Tools, Skills, Cowork. Was Claude wirklich kann.

CEO-GPT · moritzmaaker.com · Mai 2026

Moritz Maaker · AI Builder & Co-Founder CEO-GPT

INTRO

Warum Claude, warum jetzt

Claude ist nicht das lauteste Modell. Es ist das Modell, mit dem ich täglich arbeite. Dieses PDF zeigt, was im Mai 2026 wirklich drin steckt: drei Modelle, drei Welten, ein Skills-System, das alles zusammenhält.

Anthropic ist der Hersteller hinter Claude. Sitz in San Francisco, gegründet 2021 von ehemaligen OpenAI-Mitarbeitern. Die Positionierung war von Anfang an klar: Safety, Reasoning, lange Kontexte. Nicht der Hype-Wettlauf um die meisten Features pro Quartal, sondern das langweiligere Spiel von Modellen, denen man Arbeit übergeben kann.

Wenn du dir gerade einen KI-Mitarbeiter aufbauen willst, ist Claude einer der ernsthaftesten Bausteine. Drei Gründe.

1. **Lange Kontexte.** Opus 4.7 und Sonnet 4.6 verarbeiten je eine Million Tokens. Das sind grob 750.000 Wörter oder ein mittleres Sachbuch in einem einzigen Prompt.[1]
2. **Agentisches Arbeiten.** Claude Code ist seit Ende 2024 die ernsthafteste CLI für AI-gestützte Entwicklung. Skills, Sub-Agents, Hooks, MCP. Alles zusammen, lokal auf deinem Rechner.[6]
3. **Connectors statt Bastelei.** Über 375 Integrationen via Model Context Protocol direkt in claude.ai. Gmail, Drive, Notion, Slack, GitHub. Kein eigener Server nötig.[20]

Dieses PDF ist mein Praxis-Breakdown. Ich nutze Claude seit über einem Jahr täglich für mein eigenes Setup: Content-Pipeline, Daily-Briefs, Kundenrecherche, Code, Voice-Bot. Was hier steht, ist nicht aus Marketing-Slides abgeschrieben, sondern aus echter Arbeit.

Was du hier nicht findest

Kein „Claude ist die beste KI". Kein Hype. Kein Bashing von ChatGPT oder Gemini. Sondern eine Karte des Systems, mit der du Entscheidungen treffen kannst.

KAPITEL 01

Die Modell-Familie

Drei aktive Modelle, klar getrennt nach Job. Opus für die schweren Brocken, Sonnet als Dauerläufer, Haiku für Tempo und niedrige Kosten.

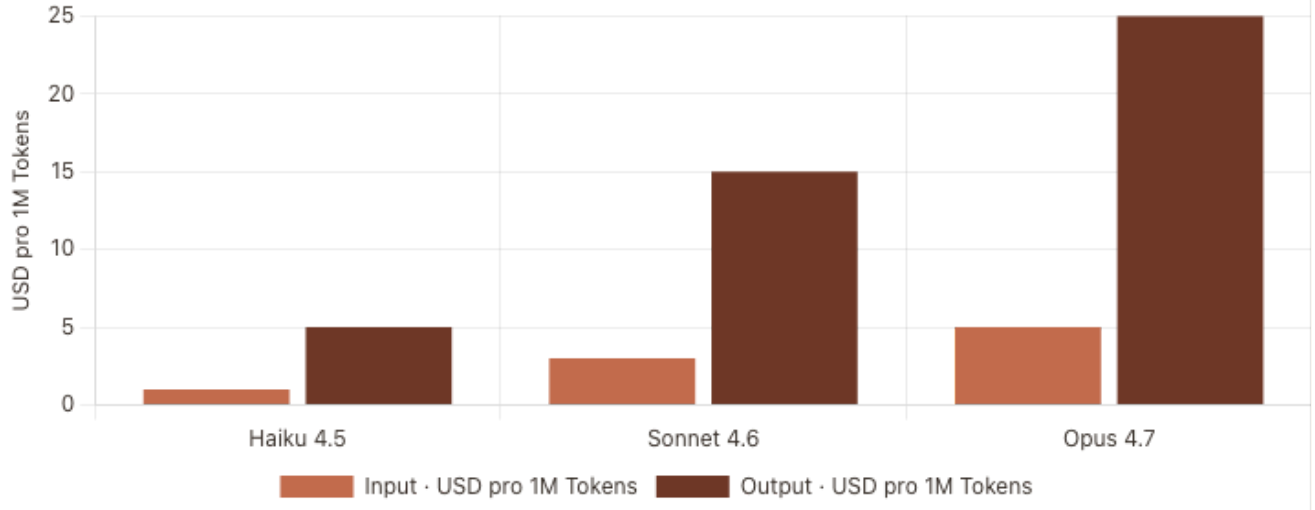
Aktuelle Modelle im Vergleich

Modell	Stärke	Context	Output	Preis Input	Preis Output
Opus 4.7	Komplexes Reasoning, agentic Coding	1 Mio. Tokens	128k Tokens	5 USD / 1M	25 USD / 1M
Sonnet 4.6	Bestes Verhältnis Speed zu Intelligenz	1 Mio. Tokens	64k Tokens	3 USD / 1M	15 USD / 1M
Haiku 4.5	Schnellstes Modell, fast Frontier-Niveau	200k Tokens	64k Tokens	1 USD / 1M	5 USD / 1M

Quelle: Anthropic Models Overview, Stand Mai 2026.[1]

Preise im direkten Vergleich

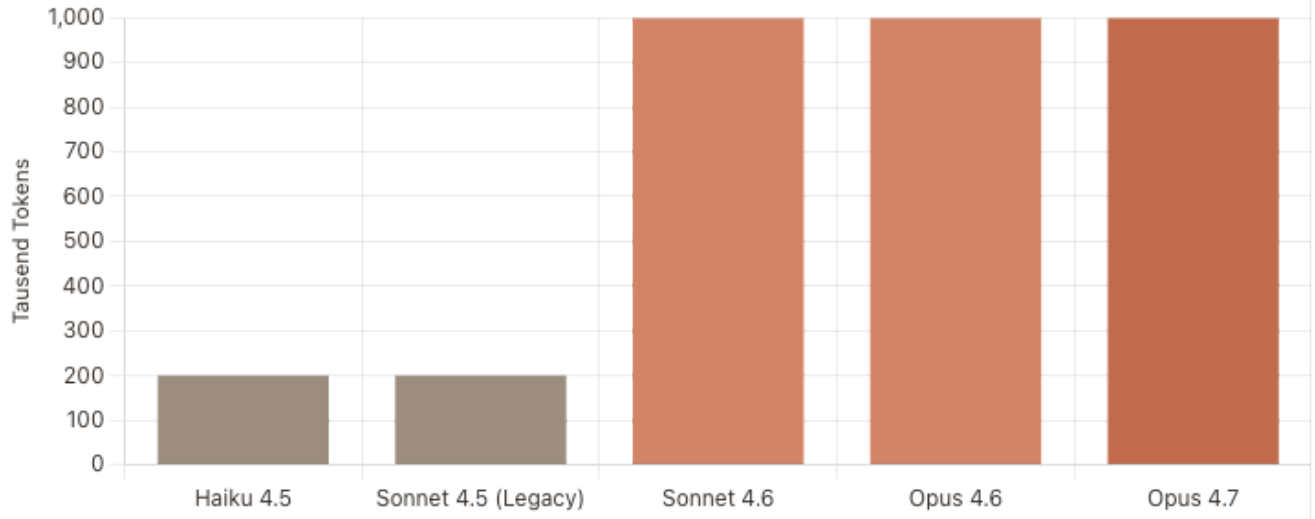
Was kostet eine Million Tokens, Input und Output



Quelle: Anthropic Models Overview, Mai 2026.[1]

Context Window im Vergleich

Tausend Tokens pro Modell, aktuelle Generation plus Sonnet 4.5 als Referenz



Quelle: Anthropic Models Overview, Mai 2026.[1]

Wann was nutzen

Opus 4.7

Anthropics aktuelles Flaggschiff. Generally available seit April 2026.[2] Stärken: agentic Coding (laut Anthropic „step-change improvement" gegenüber 4.6), Long-Context-Reasoning, präzises Instruction-Following. In meinem Setup kommt Opus immer dann zum Einsatz, wenn ein Task länger als zwei Stunden dauert oder mehr als zehn Files berührt. Strategie-Calls, Boardroom-Sessions, längere Recherchen, alles wo das letzte Quäntchen Qualität zählt.

Sonnet 4.6

Der Allrounder. Default-Modell auf den meisten Claude-Plänen. Seit Februar 2026 verfügbar.[3] Eine Million Tokens Context, deutlich schneller als Opus, ein Drittel des Preises. Wenn ich Content schreibe, Inbox triagiere oder ein Skript debugge, läuft Sonnet. In Anthropics eigenen Tests wurde Sonnet 4.6 in 70 Prozent der Fälle gegenüber Sonnet 4.5 bevorzugt und schlug sogar Opus 4.5 in 59 Prozent der direkten Vergleiche.[3]

Haiku 4.5

Seit Oktober 2025 die schnelle Alternative.[4] Kostet einen Dollar Input, fünf Dollar Output pro Million Tokens. Anthropic schreibt: Haiku 4.5 erreicht ähnliche Coding-Qualität wie Sonnet 4 zu einem Drittel des Preises und doppelter Geschwindigkeit. Bei Computer-Use schlägt es sogar Sonnet 4. In meinem Setup ist Haiku der Router: er entscheidet, welcher Sub-Agent welchen Task übernimmt. Das spart Sonnet- und Opus-Kosten für das, was wirklich Intelligenz braucht.

Legacy-Modelle

Opus 4.6, Opus 4.5, Sonnet 4.5, Opus 4.1 sind weiter erreichbar.[1] Sonnet 4 und Opus 4 sind deprecated und werden am 15. Juni 2026 abgeschaltet. Wenn deine App noch auf den alten Endpoints läuft: jetzt migrieren, nicht im Juni.

Mein Standard-Stack

Haiku als Router und für Kurz-Triage. Sonnet 4.6 als Dauerläufer. Opus 4.7 für die zwei oder drei wirklich harten Tasks pro Tag. Über den ganzen Monat liegt der Mix grob bei 60 Prozent Sonnet, 25 Prozent Haiku, 15 Prozent Opus.

KAPITEL 02

Die drei Welten

Claude lebt in drei klar getrennten Produkten. Wer den Unterschied nicht versteht, kauft das falsche Abo oder baut das falsche System. Hier ist die Karte.

Welt 1 · Claude.ai

Die Chat-UI im Browser unter claude.ai. Das ist der Einstieg für die meisten. Free, Pro (17 USD im Monat bei Jahres-Abo, 20 USD monatlich) oder Max ab 100 USD im Monat.[15]

Was Claude.ai kann

- **Projects.** Eigene Workspaces mit Custom Instructions und Files. Wie ein themenspezifischer GPT, nur ohne das Marketplace-Theater.
- **Files & Memory.** Hochgeladene Dokumente bleiben für die ganze Konversation im Kontext. Memory speichert über Sessions hinweg, was du Claude beibringst.
- **Artifacts.** Code, HTML, SVG, React-Komponenten werden direkt im Chat als Vorschau gerendert und können iteriert werden.
- **Research.** Mehrstufige Recherche-Tasks, in denen Claude selbstständig Quellen abruft, vergleicht und einen Report schreibt.
- **Skills.** Auch in claude.ai funktionieren Custom Skills (Pro, Max, Team, Enterprise, Code-Execution muss aktiviert sein).[11]
- **Connectors.** Über 375 Integrationen direkt im Chat. Dazu gleich mehr.

Wann du Claude.ai nutzt

Recherche, Briefings, Strategie-Sessions, Schreiben, Bild-Analyse, alles wo du im Browser oder auf dem Handy schnell reagieren willst. Wenn du noch keinen technischen Stack hast, ist das dein Startpunkt.

Welt 2 · Claude Code

Eine eigenständige CLI plus IDE-Plugins. Installiert wird sie über `curl -fsSL https://claude.ai/install.sh | bash` oder als VS-Code-Extension. Voraussetzung ist ein Claude-Abo oder Anthropic-Console-Account.[6]

Claude Code ist nicht „Claude im Terminal". Es ist ein agentisches System mit eigenen Bausteinen, die im Chat-UI nicht existieren.

Feature	Was es macht
CLAUDE.md	Markdown-Datei im Projekt-Root. Wird automatisch als System-Prompt geladen. Hier stehen Coding-Standards, Architektur, Konventionen.
Skills	Modulare Capabilities in `.claude/skills/<name>/SKILL.md`. Werden automatisch geladen, wenn die Beschreibung zur Aufgabe passt.
Sub-Agents	Spezialisierte Worker mit eigenem Context-Window und Tool-Set. Gut für Recherche, lange Suchen, Validierung.[7]
Hooks	Shell-Commands oder HTTP-Endpoints, die bei Events wie PreToolUse, PostToolUse, SessionStart feuern.[8]
MCP-Server	Bridge zu externen Tools: Datenbanken, GitHub, Sentry, Notion, Stripe. Konfiguration via `.mcp.json`.[10]
Plugins	Verschnürte Bundles aus Skills, Agents, Hooks und MCP-Servern. Installierbar aus dem Community-Marketplace.[9]
Background Agents	Mehrere Sessions parallel laufen lassen und aus einem Screen überwachen.
Worktrees	Mehrere Git-Worktrees parallel öffnen, jeder mit eigener Claude-Session.
Routines	Wiederkehrende Tasks auf Anthropic-Infrastruktur scheduln (PR-Reviews, Log-Analysen).

Wann du Claude Code nutzt

Wenn du Repos hast. Wenn du automatisieren willst. Wenn du mit echten Files arbeitest und nicht mit Copy-Paste. Mein gesamtes AIOS-Setup läuft über Claude Code: Content-Pipeline, Daily-Brief, Voice-Bot, alle Skills. Sobald du einen Workflow zwei Mal manuell gemacht hast, baust du ihn als Skill.

Welt 3 · Claude API / Anthropic SDK

Die rohe Programmier-Schnittstelle für eigene Apps. SDKs gibt es offiziell für Python, TypeScript, Java, Go, Ruby. Plus Bedrock (AWS), Vertex AI (Google Cloud) und Microsoft Foundry als Cloud-Endpoints.[1]

Features, die nur über die API gehen

- **Prompt Caching.** Statische Prompt-Teile zwischenspeichern, beim Re-Use 90 Prozent Kostenreduktion. Mehr dazu in Kapitel 5.[12]
- **Batch API.** Bis zu 50 Prozent Rabatt auf Input und Output, Antwort innerhalb von 24 Stunden. Für Bulk-Jobs.
- **Files API.** Files upload und über mehrere Calls hinweg referenzieren.
- **Citations.** Claude zitiert direkt aus den hochgeladenen Quellen mit Quellenangaben.
- **Tool Use.** Native Tool-Calls, die das Modell selbst ausführt und in eine Konversation einbaut.

- **Computer Use.** Maus, Tastatur, Screenshots, autonome Desktop-Steuerung. Beta, aber funktioniert auf Opus 4.7, Opus 4.6, Sonnet 4.6, Sonnet 4.5, Haiku 4.5.[13]
- **Extended & Adaptive Thinking.** Modell denkt vor der Antwort länger nach, wenn der Task komplex ist.

Wann du die API nutzt

Wenn du eigene Apps baust. Wenn du Claude in dein Produkt integrierst. Wenn du Caching oder Batch brauchst. Für Otto-Geschäftsführer wahrscheinlich nicht direkt, aber sobald du einen Agentur-Partner hast, läuft eure interne Pipeline meistens über die API.

Welt 4 · Claude in Slack, Notion, Drive, Gmail

Die Connectors. Anthropic hat über 375 Integrationen direkt in claude.ai eingebaut.[20] Sie laufen alle über das Model Context Protocol (MCP). Du klickst einmal auf „Connect", autorisierst per OAuth, und Claude kann ab dann lesen und schreiben.

Wichtigste Connectors im Mai 2026

Kategorie	Integrationen (Auszug)
Communication	Gmail, Microsoft 365, Slack (seit Januar 2026 für Pro und Max)
Content & Docs	Google Drive, Notion, WordPress.com
Projekt-Management	Asana, Linear, Jira, Monday.com
Design	Canva, Figma
Engineering & Data	GitHub, Hex, Amplitude
Finance & Health	Stripe, Apple Health, PubMed

Quelle: Anthropic Connectors Directory, Stand Mai 2026.[20]

Connectors sind in **allen** Plänen verfügbar, auch im Free-Tier. Es fällt keine zusätzliche Gebühr an.

Connectors vs. eigene MCP-Server

Connectors sind die Fertigprodukte, die Anthropic kuratiert. Wenn du etwas Spezielles brauchst (eine eigene Datenbank, ein internes Tool), baust du dir einen MCP-Server selbst. Das geht über das offene Protokoll und ist nicht auf Claude beschränkt. ChatGPT, Cursor und VS Code sprechen ebenfalls MCP.[14]

KAPITEL 03

Skills-System und Plugins

Skills sind der wichtigste Hebel, um Claude zu deinem persönlichen Mitarbeiter zu machen. Eine Skill ist ein Ordner mit einer SKILL.md plus optionalen Hilfsdateien.

Wie eine Skill aufgebaut ist

Jede Skill braucht eine `SKILL.md` mit YAML-Frontmatter. Das sieht so aus:

SKILL.MD (BEISPIEL)

```
---
name: content
description: Master-Chat für alles rund um Content. Reels-Tracker, Caption-Reviews, DM-Funnel,
Cross-Posting.
---

# Content

Wenn der User Content-Themen anspricht, lade zuerst:
- 04-areas/content/Content-Operating-State.md
- 04-areas/content/playbooks/brand-rules.md

Wenn er nach Wettbewerber fragt:
- 05-reference/competitor-content/_daily-overview/[heute].md
```

Drei Lade-Stufen

Anthropic nennt das „Progressive Disclosure". Eine Skill wird in drei Schritten geladen:[11]

1. **Metadata (immer geladen).** Name und Beschreibung, etwa 100 Tokens pro Skill. Claude weiß: diese Skill existiert.
2. **Instructions (wenn getriggert).** Der Body der SKILL.md, unter 5.000 Tokens. Wird über Bash gelesen, wenn die Description zum Task passt.
3. **Resources (on demand).** Bundle-Files, Scripts, Reference-Material. Beliebig groß, Skripte verbrauchen keinen Context, nur ihre Outputs.

Das ist der Trick: Du kannst dutzende Skills installiert haben, ohne dass dein Context-Window darunter leidet.

Wo Skills funktionieren

Surface	Custom Skills	Pre-built Skills
claude.ai	Ja (Pro+, Upload als ZIP)	Ja (PowerPoint, Excel, Word, PDF)
Claude Code	Ja, filesystem-basiert	Nein
Claude API	Ja, via Skills-API	Ja (pptx, xlsx, docx, pdf)

Quelle: Anthropic Agent Skills Overview.[11]

Wichtig: Skills synchronisieren nicht zwischen den Welten. Eine Skill in claude.ai musst du separat in der API uploaden. Claude-Code-Skills sind ein eigenes Universum auf deinem Dateisystem.

Plugin-Marketplace

Plugins sind Skills plus mehr: Agents, Hooks, MCP-Server, LSP-Server, Background-Monitors, alles in einem Bundle.[9] Anthropic betreibt zwei öffentliche Marketplaces:

- **claude-plugins-official.** Anthropic-kuratiert. In jeder Claude-Code-Installation automatisch verfügbar.
- **claude-community.** Public-Submissions nach Review. Aktivierung über `/plugin marketplace add anthropics/claude-plugins-community`.[19]

Installation läuft über den `/plugin`-Command innerhalb von Claude Code. Update bei jedem Version-Bump automatisch.

Beispiel aus meinem Setup

Bei mir leben aktuell 15 Custom-Skills unter `~/Developer/AIOS/.claude/skills/`. Drei davon:

- `/content`. Master-Chat für alles, was mit Content zu tun hat. Lädt die richtigen Operating-State-Files und Brand-Regeln automatisch.
- `/greene`. Persona-Activation für Robert Greene. 48 Laws of Power, Human Nature, 33 War Strategies als Wissens-Bundle.
- `/hormozi`. Wissens-Skill aus drei Hormozi-Büchern plus 30 Videos. Triggert nur bei expliziter Anrede.

Jede Skill ist drei bis fünf Markdown-Dateien plus ein paar Helper-Scripts. Investment war jeweils ein Nachmittag. Zeitersparnis seitdem: täglich.

KAPITEL 04

Cowork · Sub-Agents und Async Tasks

Ein einzelner Claude-Chat ist eine lineare Konversation. Sub-Agents brechen das auf. Du delegierst Aufgaben an spezialisierte Worker, die parallel arbeiten und nur das Ergebnis zurückliefern.

Was ein Sub-Agent ist

Ein Sub-Agent läuft in einem eigenen Context-Window mit eigenem System-Prompt, eigenem Tool-Set und unabhängigen Permissions.[7] Claude entscheidet selbst, wann er einen Sub-Agent ruft (anhand der Description) oder du forderst es explizit an.

Der Hauptzweck ist Context-Hygiene. Wenn du dem Haupt-Chat das Lesen von 50 Files erlauben würdest, wäre das Context-Fenster nach drei Minuten zugemüllt. Stattdessen schickt der Haupt-Chat einen Sub-Agent los, der die 50 Files liest, die Essenz extrahiert und nur eine zweiseitige Zusammenfassung zurückgibt.

Vier Vorteile

- **Context preservieren.** Suchen und Logs bleiben aus dem Haupt-Chat raus.
- **Constraints enforcen.** Du kannst Tool-Zugriff pro Sub-Agent beschränken.
- **Konfiguration wiederverwenden.** User-level Sub-Agents funktionieren über Projekte hinweg.
- **Kosten kontrollieren.** Du kannst Routine-Sub-Agents auf Haiku schicken und nur das Main-Modell auf Sonnet oder Opus laufen lassen.

Background-Agents und Worktrees

Sub-Agents arbeiten innerhalb einer Session. Wenn du komplette Sessions parallel willst, nutzt du Background-Agents. Das ist Claude Codes Multi-Session-Mode: drei oder vier komplette Claude-Instanzen, jede mit eigener CLAUDE.md, eigenem Repo-Branch (Worktree), eigenem Workflow.

Beispiel: ein Worktree refactored das Backend, einer schreibt Tests, einer aktualisiert die Docs. Du orchestrierst aus einem Übersichts-Screen.

Praxis-Beispiel

Als ich diese fünf Lead-Magnet-PDFs gebaut habe, lief pro PDF ein eigener Sub-Agent. Fünf parallel, jeder mit eigenem Working-Folder und eigener Research. Das wäre in einem linearen Chat schon nach dem zweiten PDF gescheitert, weil das Context-Window übergelaufen wäre.

Routines

Wenn du Tasks zeitgesteuert laufen lassen willst, nutzt du Routines. Das sind Cron-Jobs auf Anthropic-Infrastruktur. PR-Review jeden Montag, Log-Analyse jede Nacht, Dependency-Audit jeden Freitag.[6]

Bei mir lokal läuft dasselbe Prinzip über macOS launchd: Instagram-Tracker um 06:00, Voice-Watcher alle 5 Minuten, Dashboard-Refresh alle 30 Minuten. Sieben aktive Background-Jobs, alle nutzen Claude Code im Headless-Mode.

KAPITEL 05

Memory und Context-Engineering

Der Engpass bei KI ist nicht die Technik. Es ist der Kontext. Memory, Prompt-Caching und CLAUDE.md sind die drei Hebel, um diesen Engpass aufzulösen.

CLAUDE.md als persistentes Gedächtnis

Eine CLAUDE.md im Projekt-Root wird bei jedem Claude-Code-Aufruf automatisch geladen. Hier stehen Coding-Standards, Architektur-Notizen, präferierte Libraries, To-Do-Konventionen, kurz: alles, was Claude über dein Projekt wissen muss.[6]

Bei mir umfasst die AIOS-CLAUDE.md etwa 350 Zeilen. Sie enthält Workspace-Pfade, Ordner-Struktur, Verhaltensregeln (Ehrlichkeit über Höflichkeit, keine Em-Dashes, Vault-Updates ohne Nachfrage), Session-Start- und Session-End-Routinen, situationsbasiertes Buchwissen-Mapping.

Jede Konversation startet damit. Ich muss nicht mehr erklären, wo was liegt oder welcher Schreibstil gewünscht ist. Das ist der Unterschied zwischen einem freundlichen Assistant und einem Mitarbeiter, der dein Business kennt.

Auto-Memory

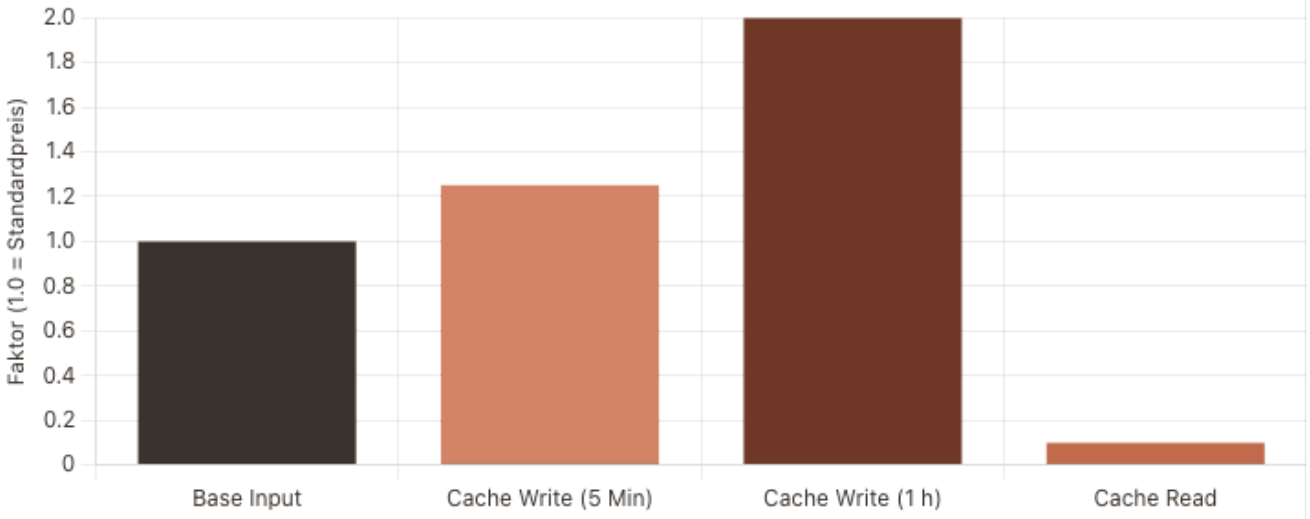
Claude Code baut zusätzlich automatisch eine Memory-Datei pro Projekt auf. Build-Commands, häufige Fehler, Workflow-Patterns landen dort, ohne dass du etwas tun musst.

Prompt-Caching

Wenn du denselben Prompt-Prefix wiederholt verwendest (System-Prompt, Tool-Definitionen, große Reference-Files), kannst du ihn cachen.[12] Die Mechanik:

Kostenfaktor Prompt-Caching

Im Vergleich zu Base-Input. Cache-Read kostet nur 10 Prozent.



Quelle: Anthropic Prompt Caching Docs, Mai 2026.[12]

- **Cache Write 5 Minuten.** 1.25x Base-Preis. Default-TTL.
- **Cache Write 1 Stunde.** 2.0x Base-Preis. Für selteneren Reuse.
- **Cache Read.** 0.1x Base-Preis. Das sind 90 Prozent Ersparnis bei jedem Hit.

Beispielrechnung: Du hast einen 100.000-Token-Prompt mit einem Knowledge-Base. Ohne Caching kostet jeder Re-Use 50 Cent (bei Opus). Mit Cache-Read sind es 5 Cent. Wenn du 1.000 Anfragen pro Tag fährst, sind das 450 Dollar gespart, jeden Tag.

Minimum-Größen

- Opus 4.7, 4.6, 4.5 und Haiku 4.5: ab 4.096 Tokens cacheable.
- Sonnet 4.6 und 4.5: ab 1.024 Tokens cacheable.

Kürzere Prompts werden nicht gecached, aber du bekommst auch keinen Fehler. Stillschweigender Skip.

1-Million-Token-Context

Opus 4.7 und Sonnet 4.6 haben ein Context-Window von einer Million Tokens.[1] Das ist mehr als die meisten Projekte je brauchen. Aber es verändert, was möglich ist:

- Komplette Codebases in einen Prompt laden statt RAG-Search bauen.
- Ein ganzes Buch lesen lassen und Q&A dagegen fahren.
- Hundert Meeting-Transkripte zusammen analysieren.
- Drei Jahre Email-Archiv zur Decision-History durchforsten.

Die Krux: Auch bei einer Million Tokens wird Caching billiger und schneller. Wer große Kontexte hat, muss Caching nutzen, sonst frisst das Budget.

Token-Budget

Praktische Regel: Halte deinen Effective-Prompt unter 30 Prozent des Modell-Limits. Bei einer Million heißt das: maximal 300.000 Tokens aktiver Kontext. Der Rest ist Puffer für Output, Reasoning, Tool-Calls. Wenn dein Kontext bei 80 Prozent liegt, fängt das Modell an, Details zu vergessen oder zu vermischen.

KAPITEL 06

Use Cases für Geschäftsführer

Drei Teile, ein Mitarbeiter. Das ist das Frame, mit dem ich CEO-GPT denke. Gehirn, Stimme, Hand. Hier sind die konkreten Use Cases pro Teil.

Gehirn · Was Claude lernen kann

Daily Brief

Jeden Morgen um 06:30 läuft bei mir ein Script, das den Tag vorbereitet. Calendar-Events, offene To-Dos aus der gestrigen Daily Note, ungelesene wichtige E-Mails, Top-Performer aus der Content-Pipeline, finanzielle Lage. Alles via Claude API mit Caching auf den persönlichen Kontext.

Strategie-Boardroom

Eine eigene Skill, die fünf Berater (CFO, Operator, Vertriebler, Mentor, Skeptiker) parallel auf eine Entscheidung loslässt. Jeder analysiert unabhängig, sie reviewen sich anonym, ein Chairman synthetisiert. Funktioniert über Sub-Agents und kostet pro Decision etwa 60 Cent. Ersetzt nicht das Bauchgefühl, aber stress-tested es.

Kontakte-CRM

Pro relevantem Kontakt eine Markdown-Datei mit Persönlichkeit, Stärken, Call-History, Pipeline-Status. Vor jedem Call lädt Claude das passende Profil und briefed mich in 30 Sekunden. Nach jedem Call wird das Profil automatisch aktualisiert mit den neuen Erkenntnissen aus dem Transkript.

Stimme · Wie Claude antwortet

Voice-Bot via Telegram

Sprachnachricht rein, Sprachnachricht raus. Telegram-Bot empfängt Audio, transkribiert, schickt an Claude, generiert Antwort, schickt sie als Audio via ElevenLabs zurück. Im Hintergrund hat Claude Zugang zu meinem ganzen Vault. Wenn ich auf dem Spaziergang frage „Wie war der Conversion-Rate letzte Woche", kommt die Antwort als Sprachnachricht in 8 Sekunden.

Content-Voiceovers

Mein Reels-System schreibt das Skript automatisch im Folder. Sobald ich `voice_status: pending` setze, generiert ein Cron-Watcher die MP3 mit meiner geklonten Stimme. Inhalt kommt von Claude, Voice von ElevenLabs. Kein manuelles Einsprechen mehr.

Hand · Was Claude tut

Email-Triage und Drafts

Gmail-Connector in claude.ai oder lokal über das Gmail-CLI. Claude geht durch die Inbox, kategorisiert, schreibt Drafts für Routine-Antworten, markiert die wichtigen für mich. Ich approve oder edit, dann raus.

Calendar-Management

Google-Calendar-Connector liest alle Termine über 30 Tage. „Wann hab ich nächste Woche zwei Stunden Block für Deep-Work" oder „Schieb den Vural-Call auf Donnerstag und schreib ihm einen Reschedule" sind Ein-Satz-Anfragen.

Content-Pipeline

Apify scraped täglich um 06:00 meine eigenen Reels plus 12 Wettbewerber. Claude verarbeitet die Daten, schreibt einen Wettbewerbs-Report, schlägt Hook-Variationen vor, baut Captions im Brand-Voice. Voice-Generation läuft async. Posting läuft manuell, der Rest automatisch.

Recherche und Reports

Eigene Deep-Research-Skill, die parallel über sieben Quellen recherchiert (Web, YouTube, Reddit, X, Academic-Paper, Podcasts, Substack). Output ist ein synthetisierter Master-Report nach drei bis fünf Minuten Laufzeit.

KAPITEL 07

15 Tipps und Tricks aus der Praxis

Was ich in einem Jahr täglicher Claude-Nutzung gelernt habe. Nicht aus Tutorials, sondern aus echten Workflows.

1. CLAUDE.md zuerst schreiben, dann Code

Bevor du das erste Mal Claude Code in einem Projekt nutzt, schreib eine CLAUDE.md. Auch wenn sie nur 50 Zeilen lang ist. Ohne dieses Briefing fängt Claude jede Session bei Null an.

2. Skills statt Mega-Prompts

Wenn du denselben Prompt zum dritten Mal copy-pastest, ist es Zeit für eine Skill. Auch wenn die Skill nur drei Zeilen Inhalt hat. Discovery passiert über die Description, also schreib sie präzise.

3. Sub-Agents für alles, was Context kostet

Recherche, Datei-Suche, Log-Analyse, Validierungs-Runs. Wenn du erwartest, dass der Output mehr als 5.000 Tokens Kontext kostet, mach es zum Sub-Agent. Dein Haupt-Chat dankt dir.

4. Prompt-Caching für Knowledge-Bases

Wenn du dieselben Reference-Files in mehreren Calls nutzt, cache sie. 90 Prozent Ersparnis bei jedem Read. Sobald dein System-Prompt über 4.096 Tokens ist (bei Opus, Haiku) oder 1.024 (bei Sonnet), lohnt es sich.

5. MCP-Server statt API-Geklatsche

Wenn du Claude mit einer externen Datenquelle verbinden willst, schau zuerst, ob ein MCP-Server existiert. Notion, GitHub, Stripe, Sentry, PostgreSQL, Airtable. Alle vorhanden. Self-Build nur, wenn nichts passt.

6. Hooks für Auto-Format und Schutz

PostToolUse-Hook auf Edit-Tool, der eslint oder ruff laufen lässt. PreToolUse-Hook, der `rm -rf` blockt. Beide brauchst du in jedem ernsthaften Projekt. Konfiguration in `.claude/settings.json`.

7. /loop und Routines für Wiederholendes

Wenn du einen Task „alle 5 Minuten" oder „jeden Montag" laufen lassen willst, nutz Routines (cloud) oder /loop (lokal). PR-Review, Log-Watch, Status-Polls.

8. Worktrees parallel laufen lassen

Git-Worktrees plus Claude-Code-Sessions pro Worktree. Drei Feature-Branches gleichzeitig in Arbeit, ohne Branch-Switching. Spart Stunden pro Woche.

9. Plan-Mode vor Edit-Mode

Bei größeren Changes lass Claude erst einen Plan schreiben, bevor er Files anfasst. `--permission-mode plan` oder explizit „mach erst einen Plan, dann warte auf mein OK". Verhindert die Hälfte der Bock-Schüsse.

10. Memory-File für persönliche Präferenzen

Anthropic baut Auto-Memory ein. Du kannst aber auch selbst eine `memory.md` oder Notes pflegen, die deine Schreib-Tics, verbotenen Wörter, Lieblings-Libraries enthält. Bei mir steht da Sachen wie „User heißt Moritz, nicht Aurélien" oder „echte Umlaute überall, niemals ae/oe/ue".

11. Connectors statt eigenem OAuth-Stack

Wenn du Claude mit Gmail, Drive, Calendar verbinden willst und keinen Code schreiben musst: nimm den Connector in claude.ai. Drei Klicks, fertig. Eigene MCP-Server nur, wenn du Workflow-spezifische Schritte brauchst.

12. Artifacts als Lieferformat

Wenn du Claude um eine Tabelle, Grafik, HTML-Vorschau oder React-Komponente bittest, frag explizit nach einem Artifact. Du bekommst eine Vorschau, kannst iterieren und am Ende exportieren. Nicht im Chat-Text wühlen.

13. Computer Use kontrolliert einsetzen

Computer Use kann jede Desktop-App steuern, aber: es ist beta, langsam und teuer (jeder Screenshot kostet Tokens). Nutz es für Tasks, die wirklich keine API haben. Vermeide es für alles, wo ein MCP-Server oder Connector existiert.

14. Batch-API für lange Runs

Bulk-Generation, Bulk-Klassifikation, Bulk-Übersetzung? Batch-API. 50 Prozent Rabatt, Antwort in unter 24 Stunden. Über Opus, Sonnet und Haiku verfügbar, mit bis zu 300k Output-Tokens pro Job für die Top-Modelle.[1]

15. Modell-Mix bewusst wählen

Routine-Triage auf Haiku. Dauer-Tasks auf Sonnet. Schwere Reasoning-Tasks auf Opus. Wer alles auf Opus fährt, zahlt das fünf- bis fünfundzwanzigfache, ohne dass die Qualität in 80 Prozent der Tasks steigt. Wer alles auf Haiku fährt, spart Geld, aber verliert bei komplexen Aufgaben.

KAPITEL 08

Versionen, Lifecycle und was bald kommt

Anthropic veröffentlicht seit 2024 in einem stabilen Quartals-Rhythmus. Hier ist die kurze Chronik plus was beim Schreiben dieses PDFs in Aussicht stand.

Modell-Chronik (Auszug)

Datum	Release	Was war neu
Mai 2024	Claude 3 Opus, Sonnet, Haiku	Erste Generation der Drei-Modell-Architektur
Juni 2024	Sonnet 3.5	Artifacts launchen, Sonnet wird Default-Modell
Oktober 2024	Computer Use	Maus- und Tastatur-Steuerung als API-Beta
Februar 2025	Sonnet 3.7	Erstes Modell mit Extended Thinking
Mai 2025	Claude 4 (Opus, Sonnet)	Agentic Coding wird Kern-Use-Case
August 2025	Opus 4.1	Verbesserung im Long-Context-Reasoning
Oktober 2025	Haiku 4.5	Near-Frontier auf Haiku-Geschwindigkeit
November 2025	Opus 4.5	SWE-Bench-State-of-the-Art[5]
Februar 2026	Sonnet 4.6	1 Mio. Context, neuer Default[3]
April 2026	Opus 4.7	Step-Change in agentic Coding, neuer Tokenizer[2]

Was deprecated wird

Anthropic kommuniziert Deprecation-Termine offen. Aktuell (Stand Mai 2026) wichtig:

- **Claude Sonnet 4** (`claude-sonnet-4-20250514`) und **Opus 4** (`claude-opus-4-20250514`) werden am **15. Juni 2026** abgeschaltet. Migration auf Sonnet 4.6 und Opus 4.7 nötig.[1]
- **Sonnet 3.7, Opus 3, Haiku 3.x** sind bereits abgeschaltet.

Praktischer Tipp: setze in deinen Apps nie auf einen Modell-Alias allein, sondern logge die genaue Version. Wenn ein Modell migriert wird, willst du das in den Antworten erkennen können.

Was in Beta läuft

- **Claude Mythos Preview.** Defensives Cybersecurity-Modell unter Project Glasswing. Invitation-only.[1]

- **Computer Use 2025-11-24.** Neuer Beta-Header für die aktuellen Opus- und Sonnet-Modelle.[13]
- **Output-300k-2026-03-24.** Extended-Output-Beta für Batch-API. Bis 300.000 Output-Tokens pro Job auf Opus 4.7, 4.6 und Sonnet 4.6.[1]

Modell-Pinning verstehen

Ab der 4.6-Generation nutzt Anthropic ein neues Naming: Modell-IDs ohne Datum (`claude-opus-4-6`) sind **trotzdem gepinnt**. Es gibt keinen Evergreen-Pointer mehr. Wenn Anthropic Opus 4.7 released, läuft `claude-opus-4-6` weiter unverändert, bis es offiziell deprecated wird. Du musst aktiv migrieren.

Das ist gut für Produktstabilität (kein nachträgliches Model-Drift) und schlecht für die, die automatisch die neuesten Verbesserungen wollen. Wer auf der Höhe bleiben will, prüft monatlich.

KAPITEL 09

Vergleich: Claude, ChatGPT, Gemini

Fair und sachlich. Jedes Modell hat einen Job, in dem es vorne liegt. Wer behauptet, eines sei „das beste", verkauft entweder ein Abo oder hat nur eines davon ernsthaft genutzt.

Stärken im Direktvergleich

Dimension	Claude	ChatGPT (OpenAI)	Gemini (Google)
Coding-Qualität	Sehr stark, agentic Coding ist die Kernkompetenz	Stark, gut bei Algorithmen und Pair-Programming	Stark, besonders gut in Google-Codebases
Long-Context	1 Mio. Tokens (Opus, Sonnet)	200k bis 400k Tokens je Modell	2 Mio. Tokens (Gemini Pro 2.5)
Agentic-Toolkit	Sub-Agents, Skills, MCP, Hooks (vollständig)	GPTs, Tools, Operator (Web-Agent)	Gemini-CLI, Notebook-LM, weniger Skill-System
Voice und Multimodal	Stark bei Bild, kein natives Voice in der UI	Voice-Mode integriert, Sora für Video	Stark bei Voice und Video, Audio-Inputs nativ
Reasoning	Adaptive und Extended Thinking, transparent	o-Modelle (o3, o4) sind State-of-the-Art	Gemini Pro Thinking, kompetitiv
Integrationen	375+ Connectors via MCP, offenes Protokoll	Native App-Library, Plugin-Store	Tiefe Google-Workspace-Integration
Konsumer-Preis	Pro 17 USD, Max ab 100 USD	Plus 20 USD, Pro 200 USD	AI Pro 20 USD, AI Ultra 250 USD
Safety-Profil	Klar dokumentierte ASL-Stufen, transparente Modell-Cards	OpenAI Spec, weniger detailliert	Google AI Principles, weniger Granular

Eigene Einschätzung Stand Mai 2026, basierend auf öffentlichen Docs und tatsächlicher Nutzung. Pricing-Angaben für ChatGPT und Gemini sind Standard-Konsumer-Tiers laut Anbieter-Websites.

Wo Claude vorne liegt

- **Agentic Coding.** Claude Code ist im Mai 2026 die ernsthafteste CLI-Lösung. Cursor und GitHub Copilot sind IDE-zentriert, Claude Code ist OS-zentriert.
- **Skills-System.** Progressive Disclosure auf Filesystem-Basis ist anderswo so klar nicht implementiert.

- **MCP-Adoption.** Anthropic hat MCP erfunden und treibt das Ökosystem. ChatGPT und Cursor sprechen MCP, aber Claude ist der nativste Client.
- **Schreibqualität.** Empirisch und in Community-Vergleichen (Reddit r/ClaudeAI, Latent Space) wird Claude für Long-Form-Writing oft bevorzugt.

Wo Claude hinten liegt

- **Voice in der UI.** ChatGPT hat einen nativen Voice-Mode, Claude nicht. Wer mit dem Modell sprechen will, baut sich das selbst (siehe meine Telegram-Bot-Lösung).
- **Bild- und Video-Generation.** Claude generiert keine Bilder oder Videos. Für DALL-E, Sora oder Veo musst du zu OpenAI oder Google.
- **App-Store-Ökosystem.** Der GPT-Store hat mehr Endkunden-Apps. Claude-Skills sind technischer und kuratierter.
- **Free-Tier.** ChatGPT-Free ist großzügiger bei der Modell-Wahl, Claude-Free ist limitierter.

Wann was nutzen

Wenn du…	Nimm
…mit Repos und CLI arbeitest	Claude (Code)
…Voice in der App brauchst	ChatGPT
…Bilder oder Videos generierst	ChatGPT (Sora) oder Gemini (Veo)
…tief im Google-Universum sitzt	Gemini
…Long-Form schreibst oder analysierst	Claude
…mehrere Agenten orchestrierst	Claude (Sub-Agents, Skills)
…Reasoning auf Benchmark-Niveau brauchst	ChatGPT (o-Modelle)

Praktischer Tipp

Du musst dich nicht entscheiden. Ein 20-Dollar-Abo bei Claude und eines bei ChatGPT kostet zusammen weniger als ein Kunde, der nicht antwortet, weil dein Vorschlag schlecht war. Ich nutze beide täglich, plus Gemini sporadisch für lange Dokumente.

QUELLENVERZEICHNIS

Quellen

Alle Quellen wurden am 26. Mai 2026 geprüft. Wenn du auf einen Link klickst, der nicht mehr funktioniert, ist meistens nur die Doku-Struktur umgezogen, der Inhalt ist über die Suche meist schnell wieder findbar.

1. Anthropic · Models Overview · platform.claude.com/docs/en/about-claude/models/overview
2. Anthropic · Claude Opus 4.7 Announcement · anthropic.com/news/claude-opus-4-7
3. Anthropic · Claude Sonnet 4.6 Announcement · anthropic.com/news/claude-sonnet-4-6
4. Anthropic · Claude Haiku 4.5 Announcement · anthropic.com/news/claude-haiku-4-5
5. Anthropic · Claude Opus 4.5 Announcement · anthropic.com/news/claude-opus-4-5
6. Claude Code · Overview · code.claude.com/docs/en/overview
7. Claude Code · Subagents · code.claude.com/docs/en/sub-agents
8. Claude Code · Hooks · code.claude.com/docs/en/hooks
9. Claude Code · Plugins · code.claude.com/docs/en/plugins
10. Claude Code · MCP · code.claude.com/docs/en/mcp
11. Anthropic · Agent Skills Overview · platform.claude.com/docs/en/agents-and-tools/agent-skills/overview
12. Anthropic · Prompt Caching · platform.claude.com/docs/en/build-with-claude/prompt-caching
13. Anthropic · Computer Use Tool · platform.claude.com/docs/en/agents-and-tools/tool-use/computer-use-tool
14. Model Context Protocol · Introduction · modelcontextprotocol.io/introduction
15. Claude · Pricing · claude.com/pricing
16. Anthropic Support · Google Workspace Integration · support.claude.com/en/articles/10167454
17. Anthropic Engineering · Equipping agents with Skills · anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills
18. Anthropic · Skills Repository auf GitHub · github.com/anthropics/skills
19. Anthropic · Community Plugin Marketplace · github.com/anthropics/claude-plugins-community
20. Claude · Connectors Directory · claude.ai/directory

MEHR DAVON? DIENSTAGS PER MAIL.

Mehr davon? Dienstags per Mail.

Jede Woche eine kurze Mail: was bei meinen Kunden funktioniert, welche neuen Skills dazukommen, ohne Bullshit.

moritzmaaker.com

Trag dich einmal ein, kriegst direkten Zugang zu allen Skills. Kein Spam, jederzeit abmelden.