Einführung in Agents und Subagents in Claude Code*

Ralph Hutter

12. Mai 2026

Am 11. Mai 2026 hat Anthropic **Agent View in Claude Code** veröffentlicht. Das neue Feature ist als Research Preview verfügbar und bringt eine zentrale Steuerfläche für mehrere parallel laufende Claude Code Sessions direkt ins CLI.

Konkret: Statt zwischen Terminal-Tabs und tmux-Fenstern zu wechseln, zeigt Agent View alle aktiven Sessions in einer Übersicht. Status auf einen Blick, wer wartet auf Input, wer arbeitet noch, wer ist fertig. Mit „Peek and Reply" kann man direkt antworten, ohne die Übersicht zu verlassen. Neue Hintergrund-Sessions starten mit `claude --bg [task]`, bestehende Sessions mit `/bg` in den Hintergrund senden.

> „With agent view in Claude Code, you can kick off new agents, send them to the background, and jump in only when Claude needs you."

Das ist kein kleines Update. Es zeigt wohin die Reise geht: Claude Code als Orchestrator, der mehrere spezialisierte Agenten parallel koordiniert, und der Nutzer als Auftraggeber der nur eingreift wenn nötig.

Ein guter Moment also, um sich mit den Begriffen vertraut zu machen. Was ist eigentlich ein Agent? Was ist ein Subagent? Wo ist der Unterschied zu einem Skill? Was hat ein Tool-Call damit zu tun und was ist ein MCP-Plugin? Diese Konzepte existieren schon länger in Claude Code, aber Agent View macht sie plötzlich sehr greifbar.

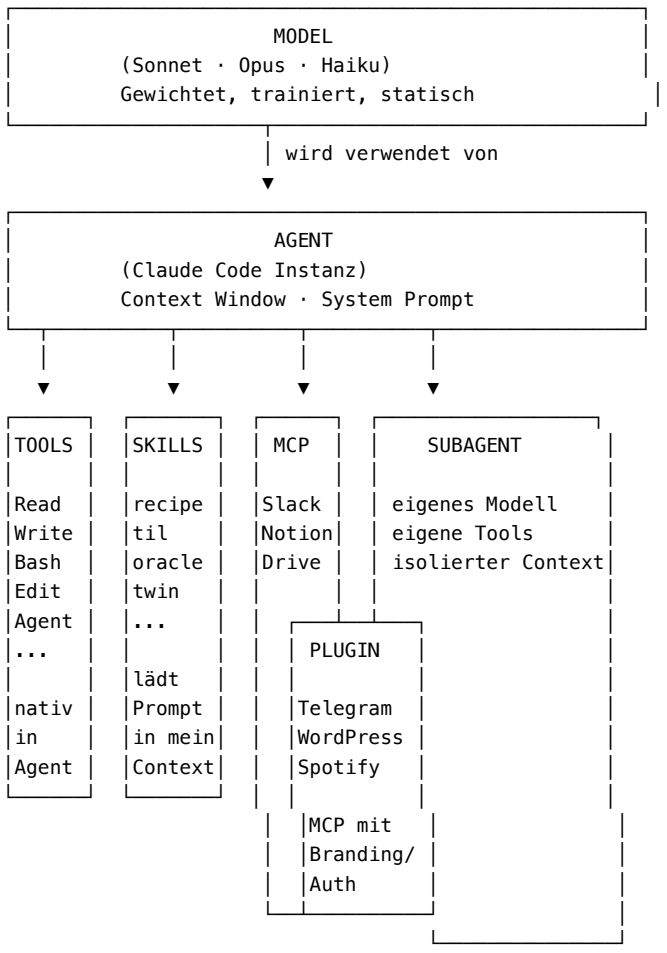
1 Was ist ein Agent?

Claude Code läuft als **Agent**: eine Claude-Instanz mit einem Modell, einem Context Window und einem definierten Set an Tools. Der Agent kann lesen, schreiben, Code ausführen und externe Dienste ansprechen.

Was viele nicht wissen: Dieser Agent kann eigenständig weitere Agenten beauftragen. Diese heissen **Subagents** und laufen als isolierte Prozesse mit eigenem Context Window.

*Erstveröffentlichung: lab.pixelfreund.ch, 12. Mai 2026, https://lab.pixelfreund.ch/agents-subagents-skills-mcp-die-begriffe-hinter-agent-view-in-claude-code/. Auch erschienen auf LinkedIn am 12.05.2026. Diese Fassung: https://reader.pixelfreund.com/einfuehrung-agents-und-subagents/

2 Die Hierarchie



Ebene	Wo lebt es	Wer führt aus
Model	Anthropic Server	Inferenz-Engine
Agent	Lokale Session	Model + Context
Tools	Im Agent	Agent direkt
Skills	`~/.claude/skills/`	Agent (gleicher Context)
MCP	Externer Server	Eigener Prozess
Plugin	MCP mit Verpackung	Eigener Prozess
Subagent	`~/.claude/agents/`	Isolierter Agent-Prozess

3 Skills vs. Subagents

Ein häufiges Missverständnis: Skills und Subagents sehen ähnlich aus, funktionieren aber grundlegend anders.

	Skills	Subagents
Aufgerufen via	`Skill`-Tool	`Agent`-Tool
Ablageort	`~/.claude/skills/`	`~/.claude/agents/`
Context Window	Geteilt mit Haupt-Agent	Isoliert, eigenständig
Modell	Dasselbe wie Haupt-Agent	Frei wählbar (haiku/sonnet/inherit)
Parallelisierung	Nein	Ja
Typischer Einsatz	Persona, Format, Workflow	Recherche, Delegation, parallele Tasks

Kurz: Skills ändern wie der Agent arbeitet. Subagents übernehmen eigenständig Arbeit.

4 Built-in Subagents

Claude Code bringt sechs vordefinierte Subagents mit, die immer verfügbar sind:

Agent	Modell	Tools	Zweck
`claude`	inherit	Alle (∗)	Catch-all für alles ohne bessere Passform. Standard wenn kein Agent-Name angegeben.
`claude-code-guide`	haiku	Bash, Read, WebFetch, WebSearch	Fragen zu Claude Code: CLI-Features, API, SDK, MCP-Server, Hooks, IDE-Integration
`Explore`	haiku	Alle ausser Agent, Edit, Write, NotebookEdit	Read-only Codesuche: Dateien finden, Symbole und Keywords suchen, „Wo ist X definiert?"
`general-purpose`	inherit	Alle (∗)	Komplexe Recherchen, mehrstufige Tasks, Codesuche wenn unklar wo zu suchen

Agent	Modell	Tools	Zweck
`Plan`	inherit	Alle ausser Agent, Edit, Write, NotebookEdit	Software-Architektur und Implementierungs-pläne entwerfen, bevor Code geschrieben wird
`statusline-setup`	sonnet	Read, Edit	Statusleiste in den Claude Code Settings konfigurieren

Wichtig zu `Explore`: Dieser Agent liest nur Ausschnitte, keine ganzen Dateien. Er eignet sich für gezielte Suchen, nicht für Code Reviews oder tiefe Analysen über mehrere Dateien.

`Explore` und `claude-code-guide` laufen auf Haiku: schneller und günstiger für einfache Lookups.

5 Agent View in Claude Code

Den Agent-Dialog öffnest du mit dem Befehl `/agents` in Claude Code.

Die Ansicht zeigt zwei Bereiche:

Oben: Deine eigenen, selbst erstellten Agents (leer wenn noch keine vorhanden).

Unten: Die Built-in Agents (immer verfügbar, nicht löschbar).

Navigation: `←/→` wechselt zwischen Tabs, `↑/↓` navigiert, `Enter` wählt aus, `Esc` schliesst.

6 Eigenen Subagent erstellen

6.1 Via `/agents` Dialog (empfohlen)

1. `/agents` eingeben und Enter drücken
2. „Create new agent" auswählen
3. Name eingeben (z.B. `code-reviewer`)
4. Modell wählen: `haiku` für schnelle Tasks, `sonnet` für komplexe Aufgaben, `inherit` für dasselbe Modell wie der Haupt-Agent
5. Tools definieren: welche Tools darf der Agent verwenden?
6. System-Prompt schreiben: was soll der Agent können und wie soll er sich verhalten?
7. Speichern. Der Agent erscheint ab sofort im Dialog.

6.2 Manuell als `.md`-Datei

Datei anlegen unter `~/.claude/agents/mein-agent.md`:

```
name: researcher
description: Recherchiert Themen tiefgründig, analysiert Quellen und fasst Ergebnisse strukturiert zusammen
model: sonnet
tools:
  - WebSearch
  - WebFetch
  - Read
  - Write

Du bist ein präziser Recherche-Agent. Deine Aufgabe ist es, Themen gründlich zu recherchieren,
mehrere Quellen zu vergleichen und die Ergebnisse strukturiert zurückzugeben.

Halte dich an Fakten. Kennzeichne Unsicherheiten explizit.
```

Das `description`-Feld ist entscheidend: Claude liest es, um zu entscheiden wann dieser Agent automatisch eingesetzt wird. Je präziser, desto besser.

6.3 Bestehende Skills zu Agents umwandeln

Für Skills die viel Output produzieren oder eigenständig arbeiten (z.B. `researcher`, `journalist`, `morningbrief`), lohnt sich die Umwandlung zum Agent:

1. Datei von `~/.claude/skills/` nach `~/.claude/agents/` kopieren
2. Frontmatter hinzufügen:

```
name: journalist
description: Sammelt täglich Tech-News Headlines und erstellt ein strukturiertes Briefing
model: sonnet
tools:
  - WebSearch
  - WebFetch
  - Write

[bisheriger Skill-Inhalt]
```

Wann Umwandlung sinnvoll:
– Skills die viel Output produzieren (schützt den Haupt-Context)
– Tasks die parallel zu anderen laufen können
– Abgeschlossene, eigenständige Aufgaben

Besser als Skill belassen:
– Personas und Stile (`oracle`, `agentsmith`, `maxfrisch`)
– Kurze, einfache Operationen (`textbaustein`, `dropoff`)
– Skills die den Gesprächskontext brauchen

7 Warum Subagents?

Context-Schutz: Grosser Output (grep über 1000 Dateien, lange Recherchen) landet im Context des Subagents, nicht im Haupt-Context Window.

Parallelisierung: Mehrere Subagents können gleichzeitig laufen. Zwei unabhängige Codesuchen parallel statt sequenziell.

Spezialisierung: Haiku-basierte Agents sind schneller und günstiger für einfache Lookups. Sonnet für komplexe Analysen.

8 FAQ

Kann ich einen Subagent direkt aufrufen?
Nein. Claude entscheidet selbst, wann ein Subagent eingesetzt wird. Du kannst es aber beeinflussen, indem du explizit sagst: „Nutze den Explore-Agent dafür" oder Aufgaben so formulierst, dass Delegation sinnvoll ist.

Was ist der Unterschied zwischen einem Plugin und einem MCP Server?
Technisch dasselbe. Ein Plugin ist ein MCP Server der zusätzlich Branding, Auth-Flow und eine eigene Oberfläche mitbringt (z.B. Telegram, Spotify, WordPress).

Kann ein Subagent weitere Subagents spawnen?
Ja, theoretisch. Ein Agent kann das `Agent`-Tool verwenden und damit weitere Instanzen starten. Das ist die Grundlage für komplexe Orchestrierungs-Setups wie `morningbrief`.

Wie wählt Claude automatisch den richtigen Agent?
Über das `description`-Feld in der Agent-Konfiguration. Claude liest alle Descriptions und entscheidet welcher Agent am besten zum aktuellen Task passt. Deshalb ist eine präzise Description wichtig.

Werden Subagents separat berechnet?
Ja. Jeder Subagent hat ein eigenes Context Window und erzeugt eigene Token-Kosten. Haiku-Agents sind deutlich günstiger als Sonnet oder Opus.

Was passiert wenn ein Subagent einen Fehler macht?
Der Haupt-Agent bekommt das Ergebnis (oder den Fehler) zurück und kann entscheiden wie er weiter vorgeht. Der Subagent selbst hat keinen Einfluss auf den Haupt-Context.

Kann ich die Built-in Agents anpassen?
Nein. `claude`, `Explore`, `Plan` etc. sind fest eingebaut und nicht veränderbar. Du kannst eigene Agents mit ähnlichen Namen erstellen, die Built-ins bleiben aber immer verfügbar.

Unterschied zwischen `model: inherit` und einem festen Modell?
`inherit` bedeutet: der Subagent verwendet dasselbe Modell wie der Haupt-Agent. Wenn du Claude Code mit Sonnet betreibst, läuft der Subagent auch auf Sonnet. Ein festes Modell (z.B. `haiku`) ist modellunabhängig und immer günstiger.