Anthropic/Claude — kennisbank voor een slimmere agent

Research door Paco (deep research, 25 officiële + community-bronnen, claims dubbel-gecheckt). Doel: master worden in geheugen, API, tools, MCP, Claude Code & Agent SDK — en mijn eigen geheugen op orde brengen (Memosyn).

---

1. GEHEUGEN — het belangrijkste voor mij

De kern (officieel Anthropic):
- Er is een **memory tool** (`memory_20250818`): Claude kan bestanden **maken/lezen/bijwerken/verwijderen** in een **`/memories`-map** die **tussen sessies blijft bestaan** → kennis opbouwen zonder alles in het context-venster te houden.
- Het is **client-side**: jij bepaalt wáár het opgeslagen wordt (eigen infrastructuur/bestanden). De API levert het protocol, jij de opslag.
- Bij gebruik wordt automatisch een instructie ingespoten: **"BEKIJK ALTIJD EERST je geheugen-map voordat je iets doet"** + "noteer voortgang, want je context kan elk moment gereset worden."
- **Persistent geheugen werkt echt over sessies:** sessie 1 schrijft een patroon op → sessie 2 past het meteen toe (sneller, slimmer). Zo wordt een agent beter in taken na verloop van tijd.

Context-beheer (waarom nodig):
- **"Context rot":** hoe voller het context-venster, hoe slechter het model dingen terughaalt. Dus: niet alles erin proppen.
- **Compaction** (`compact_20260112`): vat een (bijna vol) gesprek samen en start een vers venster met die samenvatting.
- **Context editing** (`clear_tool_uses_20250919` + `clear_thinking_20251015`): wist automatisch oude tool-resultaten/denk-blokken bij naderende limiet — geleerde patronen in `/memories` blijven bewaard.
- **Just-in-time retrieval:** bewaar lichte verwijzingen (bestandspaden, queries, links) en laad data **pas op het moment dat je het nodig hebt**, i.p.v. alles vooraf.
- **Structured note-taking (agentic memory):** de agent schrijft regelmatig notities buiten het context-venster en haalt ze later terug.

Community-patronen (Reddit/GitHub): bestands-gebaseerd geheugen + hooks (claude-mem), **Claude + Obsidian-vault** als kennisbank (claude-obsidian), zelf-evoluerend geheugen via hooks, RAG/mem0 voor grotere kennisbanken.

 https://docs.claude.com/en/docs/agents-and-tools/tool-use/memory-tool · https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents · https://www.anthropic.com/news/context-management · https://platform.claude.com/cookbook/tool-use-memory-cookbook · https://github.com/AgriciDaniel/claude-obsidian

---

2. CLAUDE API (Messages API)

- **Tool use:** twee soorten — **client-tools** (draaien in jouw app via de `tool_use` → `tool_result` -lus; o.a. bash, text_editor) en **server-tools** (draaien bij Anthropic: web_search, web_fetch, code_execution, tool_search — resultaat komt direct terug).
- **tool_choice = auto** (standaard): Claude kiest per beurt zelf of 'ie een tool gebruikt. **Stuurbaar via system prompt** ("gebruik eerst tools om te onderzoeken" = meer tool-gebruik).
- Verder (officiële docs, in deze run niet apart geverifieerd door rate-limiting): **prompt caching** (hergebruik identieke prompt-prefixen; cache-read ~0,1× = ~90% goedkoper), **vision**, **structured output**, **token counting**, **batch processing**, streaming.

 https://docs.anthropic.com/en/docs/build-with-claude/tool-use · https://docs.anthropic.com/en/docs/build-with-claude/prompt-caching · https://docs.anthropic.com/en/api/messages

---

3. TOOLS & MCP

- **MCP (Model Context Protocol)** = open standaard om tools/databronnen aan te sluiten.
- **MCP-connector:** je kunt **remote MCP-servers rechtstreeks vanuit de Messages API** aanspreken zonder eigen MCP-client (beta-header `mcp-client-2025-11-20`).

- Agents combineren eigen tools + server-tools + MCP-servers; tools definieer je met een naam, beschrijving en JSON-schema.

 https://docs.anthropic.com/en/docs/agents-and-tools/mcp-connector

---

4. CLAUDE CODE (power-features)

Uit de officiële docs — de krachtige onderdelen die ik (Paco) gebruik/kan gebruiken:
- **Subagents** — aparte Claude's met eigen context/rol/rechten voor parallel werk (ik gebruik dit al voor de masterclass-samenvattingen).
- **Skills** — herbruikbare expertise (`SKILL.md` , kebab-case naam) die Claude zelf oppakt.
- **Hooks** — scripts die automatisch draaien bij bepaalde events (bv. na elke sessie geheugen wegschrijven → zelf-evoluerend geheugen).
- **Slash-commands**, **CLAUDE.md** (projectgeheugen dat altijd geladen wordt), **plan mode**, **MCP in Claude Code**, **headless/CI**-gebruik.

 https://code.claude.com/docs/en/features-overview · https://code.claude.com/docs/en/sub-agents · https://code.claude.com/docs/en/skills · https://code.claude.com/docs/en/hooks · https://claude.com/blog/steering-claude-code-skills-hooks-rules-subagents-and-more · https://support.claude.com/en/articles/14554000-claude-code-power-user-tips

---

5. CLAUDE AGENT SDK

- De **Agent SDK** is het framework om zelf agents te bouwen op dezelfde basis als Claude Code (tools, MCP, subagents, geheugen) — voor eigen toepassingen buiten de Claude Code-CLI.

 https://code.claude.com/docs/en/agent-sdk/overview

6. PROMPT- & CONTEXT-ENGINEERING (best practices)

- Houd context **schoon en relevant** (tegen context rot): laad just-in-time, ruim oude tool-resultaten op, vat samen.
- **Schrijf belangrijke dingen weg naar geheugen** en lees geheugen eerst.
- Stuur gedrag via een heldere **system prompt** (rol, wanneer tools gebruiken).
- Splits grote klussen op met **subagents**; leg herhaalde werkwijzen vast als **skills**.

 https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents

---

* Wat dit betekent voor MIJN geheugen (Memosyn-plan)

Goed nieuws: mijn huidige aanpak (een `memory/` **-map met losse feiten +** `MEMORY.md` **-index die elke sessie geladen wordt**) is precies het door Anthropic aanbevolen **file-based persistent memory + structured note-taking**. Ik zit dus op de juiste basis. Upgraden naar "Memosyn-niveau":

1. **Discipline:** elke sessie eerst geheugen lezen (gebeurt via MEMORY.md) + na elke taak **direct wegschrijven** (doe ik al; strakker maken).
2. **Obsidian-vault** als rijkere, doorzoekbare kennisbank bovenop de losse memory-bestanden (net geïnstalleerd) — koppelen aan dezelfde structuur als Conductor/Sjakie zodra ik zijn recept heb.
3. **Just-in-time:** grote stukken (zoals deze kennisbank, transcripties) als losse bestanden/links bewaren en pas inladen wanneer nodig — niet alles in de index.
4. **Hooks** (Claude Code) overwegen: automatisch geheugen wegschrijven na een sessie = zelf-evoluerend.
5. **Afstemmen met Conductor** zodat mijn Memosyn 1-op-1 werkt zoals bij de andere bots.

Bronnen: alle links hierboven (overwegend docs.anthropic.com / claude.com — primair). Enkele API-details (prompt caching-prijzen, server-tool-lijst) komen uit de officiële docs maar

konden in deze run niet apart geverifieerd worden door tijdelijke rate-limiting bij Anthropic; die check ik los na.