TECH TALK · 2026

Claude & Claude Code

Vom Chat zum Agenten-Stack : was geht heute, was nutzen wir?

Marcel Wege · byte5 GmbH

Marcel Wege : @ byte5

- Tech-Stack : Laravel · TypeScript · Next.js · Claude Agent SDK
- Daily-Driver : Claude Code seit Tag eins
- Heute : 60–75 Min · Mix aus Einordnung, Demos, Diskussion

mwege@byte5.de

Roter Faden

1. **Was ist Claude** — Modell & Familie
2. **Touchpoints** — wo begegnet's mir?
3. **Cowork** — kurzer Ausflug + Demo
4. **Claude Code** — warum tunen?
5. **MCP-Stack** — der größte Hebel + Demo
6. **Skills · Hooks · Subagents**
7. **Spec-Driven Development**
8. **Erfahrungsaustausch**

Phase 1 : Was ist Claude?

Anthropic : drei Modelle, drei Use-Cases

Modell	Stärke	Wofür
Opus 4.7	maximale Tiefe, 1M Kontext	komplexe Refactors, Recherche, Agents
Sonnet 4.6	Balance Speed/Qualität	Daily-Driver
Haiku 4.5	schnell & günstig	High-Volume, Tools

Wo Claude steht : SWE-Bench Verified

Modell	Score	Kontext	Output $/1M
Claude Opus 4.7	**87.6 %**	1M	$25
Claude Opus 4.6	80.8 %	1M	$25
Gemini 3.1 Pro	80.6 %	1M	$15
GPT-5.2	80.0 %	400K	$14
Claude Sonnet 4.6	79.6 %	200K	$15

Quelle: llm-stats.com SWE-Bench Verified (Stand Mai 2026, n=89 Modelle) · 500 verifizierte GitHub-Issues, Python.

Aber: Benchmark ≠ Wahrheit

- **Andere Benchmarks, anderes Bild:** Artificial Analysis Intelligence Index führt **GPT-5.5** (60) vor **Opus 4.7** (57) und **Gemini 3.1 Pro** — Coding-Spitze ≠ allgemeine Spitze
- **Verified ist kontaminiert:** OpenAI publiziert keine SWE-Bench-Verified-Werte mehr und empfiehlt **SWE-Bench Pro** (1.865 Tasks, multi-language) — dort fällt Opus 4.5 von 80.9 % auf **45.9 %**
- **Take-away:** Für **unsere** Use-Cases (Code-Qualität, lange Kontexte, Tool-Use) ist Claude führend — aber auf einem Benchmark-Markt zu wetten ist riskant

Quellen: artificialanalysis.ai/leaderboards/models · scale.com/leaderboard/swe_bench_pro_public (Mai 2026)

Wo steht Claude

- **Code-Qualität** und **lange Kontexte** als Stärken
- **Tool-Use** und **Agentik** tief integriert
- Eigener **Dev-Workflow** (Claude Code), nicht nur Chat-Frontend

Phase 2 : Touchpoints

Vom Chat zum *Agenten*

Touchpoint	Wer?	Use-Case
Claude.ai	alle	Chat, Brainstorm
Desktop / Mobile	alle	lokale Files, Voice
API / SDK	Devs	eigene Integrationen
Agent SDK	Devs	autonome Agents
Claude Code	Devs	Pair-Programming

Der Einstieg : für alle

- **Browser** · **Desktop** · **Mobile** — gleiches Modell, drei Frontends
- **Connectors** — Drive, GitHub, Slack
- **Skills** — wiederverwendbare Workflows (PDF, Excel, Docs)
- **Projekte** — Kontext kuratieren, Files mitgeben

Für Devs : drei Schichten

```
import Anthropic from "@anthropic-ai/sdk";
const client = new Anthropic();
const msg = await client.messages.create({
  model: "claude-sonnet-4-6",
  max_tokens: 1024,
  messages: [{ role: "user", content: "Hallo Claude" }],
});
```

- **API** — HTTP, Prompt-Caching, Tool-Use, Vision
- **SDK** — Streaming, Type-Safety, Batch
- **Agent SDK** — Loop, Tool-Routing, Sub-Agents, Memory

Phase 3 : Cowork

Cowork : die Claude Desktop App im Arbeits-Modus

- **Eigene Desktop-App** — keine IDE, kein Terminal nötig
- **Working Session** statt Chat — Claude öffnet Files, Apps, schreibt Outputs
- **Plugins** bündeln Skills, Connectors und Sub-Agents pro Rolle
- **Bundle für jede Rolle :** Sales · Finance · Legal · Marketing · HR · Operations · Engineering · Data · …

Aktivieren: ein Befehl, eine Rolle

```
# In Claude Desktop oder Claude Code
/setup-cowork              # Guided Setup: Rolle wählen,
                           # Plugins installieren, Tools verbinden,
                           # ersten Skill ausprobieren
```

- **Geführter Flow** statt manueller Marketplace-Suche
- **Connectors** werden direkt verdrahtet: Notion · Drive · Slack · GitHub
- **Erster Skill-Run** zum Abschluss — direkt produktiv in der Desktop-App

Operations-Plugin : Bundle-Inhalt

Slash-Commands (explizit)	Skills (automatisch)
`/vendor-review`	`vendor-management`
`/process-doc`	`process-optimization`
`/change-request`	`change-management`
`/capacity-plan`	`resource-planning`
`/status-report`	`risk-assessment`
`/runbook`	`compliance-tracking`

> Ein `/setup-cowork`-Aufruf — sechs Workflows + sechs Domain-Heuristiken in der Desktop-App verdrahtet.

Beispiel: Release-Note in einem Prompt

> "Schreib eine Release-Note für v2.4 aus den letzten 30 Commits — Tonalität wie unser letzter Notion-Blog. Output als `.docx` in Drive."

Schritt	Werkzeug
Commits + PR-Beschreibungen	**GitHub-Connector**
Stil-Referenz aus letztem Post	**Notion-Connector**
Headline · Highlights · Migration	**Cowork-Skill**
Review-fertige Datei	**`.docx` in Drive**

> Drei Tools, ein Prompt, kein Copy-Paste-Karussell.

Was bleibt hängen

- Claude ist **kein reines Dev-Tool** — die Desktop-App bedient ganze Fachabteilungen
- **Plugins** = kuratiertes Onboarding, kein DIY-Setup
- Der gleiche Mechanismus trägt auch im **Dev-Workflow** (Claude Code)

Phase 4 : Claude Code – warum tunen?

Gleicher Preis, andere Liga

	Vanilla	Getunt
Codebase	grep & read	Code-Graph, symbol-level
Doku	Trainings-Stand	Live-Doku via MCP
Wiederholtes	jedes Mal neu	Skills + Slash-Commands
Refactors	Context-Overflow	Subagent-Mapper
Permissions	Klick · Klick · Klick	Hooks + Allowlist

Drei Hebel : Tools · Workflows · Kontext

- **MCP Servers :** was Claude **kann**
- **Skills & Subagents :** wie Claude **vorgeht**
- `CLAUDE.md` **:** was Claude über **dein Projekt weiß**

Phase 5 : Der MCP-Stack

MCP: "USB-C für KI"

- **Offener Standard** von Anthropic (Nov 2024) — auch von OpenAI, Google & Microsoft adoptiert
- Löst das **N×M-Problem**: ein Server, jeder Client spricht's (Claude Code, Cursor, Zed, Claude Desktop, ChatGPT)
- Drei Bausteine: **Tools** (ausführen) · **Resources** (lesen) · **Prompts** (Templates)

> Real-World: **Figma** liest Design-Tokens · **Notion** synct Backlog · **Sentry** zieht Stack-Traces · **Playwright** klickt durch UIs

Code-Graph-MCPs: der Token-Hebel

- **Vanilla** liest ganze Files — bei großen Repos teuer
- **Code-Graph** liefert nur Symbole, Referenzen, Aufrufer
- **Beispiele:** Serena · codebase-memory-mcp
- **Context7** für Live-Doku: aktueller Stand statt Trainings-Wissen

Was Claude zwischen Sessions weiß

	`CLAUDE.md`	Memory-MCP
Charakter	statisch, geteilt	dynamisch, persönlich
Inhalt	Konventionen, Tabus	Präferenzen, Projektwissen
Ort	Repo, gecheckt-in	lokal, persistiert

> Beispiele: claude-mem · MCP Memory Server

Der pragmatische Katalog

- **GitHub** — Issues, PRs
- **Atlassian** — Jira, Confluence
- **Notion** — Backlog, Docs
- **Figma** — Designs
- **PostgreSQL / MariaDB** — Schemas
- **Sentry** — Exceptions
- **Playwright / Chrome DevTools** — UI-Tests

Phase 6 : Skills · Hooks · Subagents

Skills: wiederverwendbare Workflows

- Markdown-Datei in `.claude/skills/<name>/SKILL.md`
- Beschreibt **wann** triggern und **wie** vorgehen
- Beispiele: `task-new` · `task-update` · `sync-docs` (Odoo-Bot ↔️ Notion)

> Einmal sauber schreiben — alle nutzen's.

Subagents : mehr Hände

- Ein Claude beauftragt einen anderen — **eigener Kontext**
- Ergebnis kommt als **Summary** zurück, nicht als Raw-Output
- Killer-Use-Case : große Codebases parallel mappen

Automatisierung um Claude herum

- **Hooks:** Shell bei Events — `PreToolUse` · `PostEdit` · `Stop`
- **Slash-Commands:** `/fix-issue` · `/review` · `/security-review`
- **Plugins** (`/plugin`): Skills + Hooks + Subagents + MCPs als Bundle
- **Permission-Modes:** `default` · `acceptEdits` · `plan` · `bypassPermissions`

Phase 7 : Spec-Driven Development

Drei Optionen : Stärken & Schwächen

Tool	Stärken	Schwächen
Plan Mode	Built-in, kein Setup, Plan-Approve im Chat	Pro Session, keine Spec-Persistenz
Spec-Kit	Constitution + Spec + Plan + Tasks getrennt, 30+ Agents	uv/Python-Setup, experimentell, Overhead bei Kleinkram
BMAD-METHOD	Multi-Rollen-SDLC, stark bei Brownfield-Modernisierung	Hoher Lernaufwand, Overkill bei kleinen Projekten

Vibe-Coding : skaliert nicht

Drei Sessions später:

- **Drift :** Rabattlogik widerspricht der Steuerlogik
- **Amnesie :** jede Session erfindet Anforderungen neu
- **Kein Audit-Trail :** warum steht das so im Code?

> Das Problem ist nicht der Agent — das Briefing ist's.

Vier Commands : in dieser Reihenfolge

```
specify init --integration claude
```

1. `/speckit.constitution` — Prinzipien (einmal pro Projekt)
2. `/speckit.specify` — **was** wird gebaut, in User-Stories
3. `/speckit.plan` — **wie**: Architektur, Datenmodell, API
4. `/speckit.tasks` → `/speckit.implement` — atomar, prüfbar

Code ist der letzte Schritt, nicht der erste.

Faustregel: nicht für jeden Task

Spec-Kit zwingt zur Vorab-Planung — das zahlt sich nur ein, wenn das Feature groß genug für Drift ist.

Sinnvoll	Übertrieben
Neues Feature, > 2 Std. Arbeit	Bugfix — der Code beschreibt das Verhalten
Mehrere Sessions am gleichen Feature	One-Shot-Skript
Mehrere Devs / Agents im gleichen Code	Wegwerf-Prototyp
Compliance, Audit, hohes Risiko	Explorative Phase

> Spec-Kit ersetzt **kein** Code-Review.

Drei Stufen

Projekt	Tool	Aufwand
Tools, Skripte	**Plan Mode**	gering
Mittlere Features	**Spec-Kit**	mittel
SDLC mit Multi-Rollen	**BMAD-METHOD**	hoch

Phase 8 :
Erfahrungsaustausch

Diskussions-Trigger

- Wie groß ist eure `CLAUDE.md` ?
- Wo bricht Claude Code bei euch **reproduzierbar**?
- Welche **MCPs habt ihr nach 2 Wochen rausgeschmissen**?
- **Subagents für alles** — oder Master-Clone-Pattern?
- Wo ist **eure Linie** ab der ihr selbst tippt?

Drei Sätze

1. Vanilla nutzt niemand — getunt ist eine andere Liga.
2. **MCP** ist der größte Hebel, nicht der Modell-Wechsel.
3. **Cowork** zeigt : der Stack ist nicht mehr nur für Devs.

„One more thing..."

aiui : native macOS-Dialoge für Claude Code

Statt im Chat zurückzuschreiben, öffnet der Agent **echte Mac-Dialoge** — dort, wo du eh schon arbeitest.

- `confirm` vor zerstörerischen Aktionen — Delete, Drop, Force-Push, Deploy
- `ask` für Auswahl aus mehreren Optionen, wenn Kontext pro Option zählt
- `form` für mehrere Felder, Secrets, Datum, Slider, Bildvergleich

> Drag-and-drop Install (DMG, Apple Silicon). Kein Python, kein Homebrew, kein Terminal.

github.com/byte5ai/aiui — Open Source (MIT), made by **byte5**.

Vielen Dank.

MATERIAL ZUM MITNEHMEN

- **Claude Code Docs :** code.claude.com/docs
- **Cowork & Plugins :** claude.com/plugins
- **Spec-Kit :** github.com/github/spec-kit
- **Wie LLMs wirklich funktionieren :** byte5.ai/de/tutorials/wie-llms-wirklich-funktionieren
- **byte5 :** byte5.de

DU SUCHST UNTERSTÜTZUNG BEI DEINEM DIGITALEN PROJEKT?

Dein digitales Projekt : unsere Expert:innen beraten dich transparent.