KI-AGENT

KI-Agenten Serie • Teil 2 von 2

OpenClaw Tutorial

11. Dezember 2025 | 25 Min. Lesezeit | TECHNICAL INSIGHT

KI-Agenten Serie 第 2 講・共 2 講

1 Einführung

2 Architektur & Deploy

← 上一講

› OpenClaw noch nicht installiert? Klicken Sie hier fur die Ein-Klick-Installationsanweisungen

Key Findings

OpenClaw ist das am schnellsten wachsende Open-Source-Projekt in der Geschichte von GitHub Anfang 2026. Es handelt sich im Wesentlichen um einen KI-Agenten, der auf Ihrem Computer installiert wird und Ihren gesamten Rechner uber WhatsApp, Telegram und andere Messaging-Apps fernsteuern kann

Die Bereitstellung ist extrem vereinfacht — ein einziger Befehl genugt zur Installation, erganzend durch wenige CLI-Befehle fur die Erstkonfiguration (Cloud-Server erfordern manuelle Einrichtung, kein interaktiver Assistent). Unterstutzt werden macOS, Linux und Windows

Die Vierschichtarchitektur (Gateway → Nodes → Channels → Skills) macht KI nicht mehr nur zu einem Chat-Werkzeug, sondern zu einem vielseitigen Assistenten fur Browser-Automatisierung, geplante Aufgaben und Softwareentwicklung

Dieser Artikel durchlauft praxisnah den gesamten Prozess: Bereitstellung, Modellkonfiguration, Messaging-Kanal-Integration, Browser-Automatisierung, geplante Berichte, Claude Code Softwareentwicklung sowie die fortgeschrittenen Themen Hooks Zero-Polling und Agent Teams Multi-Agenten-Zusammenarbeit

Schritt-fur-Schritt-Navigation

I. Warum Sie sich mit autonomen KI-Agenten beschaftigen sollten

Anfang 2026 durchlauft die KI-Branche einen stillen, aber tiefgreifenden Paradigmenwechsel: **Vom Dialogmodus „Mensch fragt, KI antwortet" hin zum Agentenmodus „Mensch definiert das Ziel, KI plant und fuhrt selbstständig aus".**

In den letzten drei Jahren waren wir an folgende KI-Interaktion gewohnt: ChatGPT oder Claude offnen, eine Frage eingeben, auf eine Antwort warten und dann selbst den nachsten Schritt entscheiden. KI war ein leistungsfahiger Berater, doch die Ausfuhrung lag weiterhin beim Menschen.

KI-Agenten (AI Agents) andern diese Logik grundlegend. Ein echter KI-Agent beantwortet nicht nur Ihre Fragen — er **versteht Ihre Absicht, zerlegt sie in Teilaufgaben, setzt Werkzeuge schrittweise ein, korrigiert sich wahrend des Prozesses selbst und liefert schliesslich das Ergebnis**. Sie mussen die KI nicht mehr Schritt fur Schritt steuern, sondern sagen wie zu einem erfahrenen Assistenten: „Erledige das fur mich" — und widmen sich anderen Dingen.

OpenClaw (fruher ClawdBot / MoltBot) ist das reprasentativste Open-Source-Produkt dieses Paradigmenwechsels. Es uberschritt auf GitHub innerhalb von zwei Tagen 100.000 Sterne[1] und wurde von Mainstream-Medien wie Scientific American und CNBC breit berichtet[2] [3]. Es wurde zum meistbeachteten KI-Projekt Anfang 2026 — nicht weil die Technologie besonders innovativ ist, sondern weil es normalen Menschen erstmals wirklich die Erfahrung ermoglichte, **was es bedeutet, wenn „KI den Computer ubernimmt"**.

In unserem vorherigen Artikel «OpenClaw Einfuhrung» haben wir OpenClaw aus Teamperspektive analysiert und Kernwert sowie Risiken bewertet. Dieser Artikel konzentriert sich auf die **praktische Umsetzung** — von Grund auf, Schritt fur Schritt, durch die vollstandige Installation, Konfiguration und sechs Praxisszenarien von OpenClaw.

II. Architektur im Uberblick: Das Vierschichtdesign von OpenClaw verstehen

Bevor Sie loslegen, nehmen Sie sich zwei Minuten, um die Architektur von OpenClaw zu verstehen — das wird Ihnen helfen, im weiteren Konfigurationsprozess schneller den Uberblick zu behalten.

OpenClaw verwendet ein **Vierschichtarchitektur**-Design[3]:

- **Gateway (Kernkomponente):** Das Herzstuck des gesamten Systems, verantwortlich fur Aufgabenplanung, Speicherverwaltung und LLM-Orchestrierung. Es lauft auf `ws://127.0.0.1:18789` . Alle Anweisungen laufen letztlich im Gateway zusammen
- **Nodes (Hardwareknoten):** Zustandig fur die Interaktion mit Ihrer Computerhardware — Dateisystemoperationen, Shell-Befehlsausfuhrung, Prozessmanagement. Unterstutzt macOS, Linux, Windows WSL2 und sogar Raspberry Pi
- **Channels (Kommunikationskanale):** Verbindet uber 10 Messaging-Plattformen wie WhatsApp, Telegram, Discord, Slack und Signal, sodass Sie uber Ihre taglichen Chat-Apps der KI Anweisungen erteilen konnen
- **Skills (Fahigkeitsmodule):** Erweiterbare Funktions-Plugins — Browser-Automatisierung, Kalenderintegration, Codeausfuhrung, Blog-Uberwachung und mehr. Uber den Skills-Store konnen die Fahigkeiten der KI kontinuierlich erweitert werden

Daruber hinaus gibt es eine das gesamte System durchziehende **Memory-Schicht (Gedachtnis)**, die Ihre Gesprachskontexte und Praferenzen in Markdown-Dateien persistent speichert und OpenClaw mit zunehmender Nutzung immer besser auf Sie abstimmt.

Sobald Sie diese vier Schichten verstanden haben, wissen Sie bei jedem weiteren Konfigurationsschritt genau, „welche Schicht" Sie gerade konfigurieren.

Im Folgenden beginnt der praktische Teil

III. Systemvoraussetzungen und Vorbereitung

Bevor Sie mit der Installation beginnen, stellen Sie sicher, dass Ihre Umgebung die folgenden Anforderungen erfullt:

- **Betriebssystem:** macOS, Linux oder Windows. Cloud-Server und Raspberry Pi werden ebenfalls unterstutzt
- **Node.js:** Version ≥ 22 (uberprufen Sie mit `node --version`)
- **Arbeitsspeicher:** Mindestens 2 GB; fur Browser-Automatisierung werden 4 GB oder mehr empfohlen
- **LLM API Key:** Sie benotigen mindestens einen API-Schlussel fur ein grosses Sprachmodell — unterstutzt werden Anthropic (Claude), OpenAI (GPT) und andere, lokale Modelle konnen ebenfalls verwendet werden

Falls Sie Node.js 22 noch nicht installiert haben, konnen Sie es wie folgt schnell installieren:

🔧 Befehl ausfuhren

```
# macOS (mit Homebrew)
brew install node@22
# oder mit nvm
nvm install 22
nvm use 22
```

Copy

STEP 1 Installation und Onboarding ⏱ ca. 5 Minuten

Der Installationsprozess von OpenClaw ist extrem vereinfacht. Offnen Sie das Terminal und fuhren Sie folgenden Befehl aus[4]:

🔧 Befehl ausfuhren

```
# macOS / Linux
curl -fsSL https://openclaw.ai/install.sh | bash
# Windows (PowerShell)
iwr -useb https://openclaw.ai/install.ps1 | iex
```

Copy

Alternativ konnen Sie uber npm global installieren:

🔧 Befehl ausfuhren

```
npm i -g openclaw
```

Copy

Nach Abschluss der Installation fuhren Sie die Ersteinrichtung durch. Hinweis: Der interaktive Assistent von `openclaw onboard` benotigt ein TTY-Terminal und schlagt auf Cloud-Servern (SSH) oder in Headless-Umgebungen fehl. Verwenden Sie daher die folgenden manuellen Schritte:

🔧 Befehl ausfuhren

```
# Konfigurationsdatei initialisieren
openclaw setup
# Gateway auf lokalen Modus setzen (fur Cloud-Server erforderlich)
openclaw config set gateway.mode local
# Daemon-Dienst installieren (systemd)
openclaw daemon install
# Daemon starten
openclaw daemon start
```

Diese vier Schritte erledigen jeweils: **Standard-Konfigurationsdatei generieren → Gateway-Modus festlegen → systemd-Dienst registrieren → Hintergrund-Daemon starten**. Wenn Sie lokal (macOS / Desktop-Linux) arbeiten und ein interaktives Terminal haben, konnen Sie alternativ `openclaw onboard` fur eine Einrichtung in einem Schritt verwenden.

Nach Abschluss der Einrichtung uberprufen Sie den Gateway-Status:

🔧 Befehl ausfuhren

```
# Gateway-Status uberprufen
openclaw gateway status
```

Sie sollten eine Meldung sehen, dass das Gateway auf `http://127.0.0.1:18789/` lauft. Sie konnen auch die Control UI-Benutzeroberflache im Browser starten:

🔧 Befehl ausfuhren

```
# Web-basierte Steuerungsoberflache offnen
openclaw dashboard
```

Die Control UI bietet eine intuitive Web-Oberflache, uber die Sie direkt im Browser mit OpenClaw kommunizieren, Nachrichtenverlaufe einsehen und den Systemstatus uberprufen konnen. Wenn Sie das Gateway manuell im Vordergrund starten mochten (haufig zum Debugging verwendet):

📖 Nur als Referenz (fur Debugging)

```
# Gateway im Vordergrund starten (geeignet fur Tests und Debugging)
openclaw gateway --port 18789
```

> ✅ **Prufpunkt**
>
> Die Ausfuhrung von `openclaw gateway status` sollte zeigen, dass das Gateway auf `http://127.0.0.1:18789/` lauft. Das Offnen von `openclaw dashboard` sollte die web-basierte Steuerungsoberflache anzeigen.

STEP 2 Modellkonfiguration: Die Engine fur das KI-Gehirn wahlen ⏱ ca. 3 Minuten

Wenn Sie die Modellkonfiguration bereits wahrend des Onboarding abgeschlossen haben, konnen Sie diesen Schritt uberspringen. Fur nachtragliche Anpassungen bietet OpenClaw einen flexiblen Modellverwaltungsmechanismus.

OpenClaw enthalt kein eingebautes grosses Sprachmodell — es muss mit einem externen LLM als Reasoning-Engine verbunden werden. Modelle werden im Format `provider/model` referenziert, z. B. `anthropic/claude-sonnet-4-5` .

🔧 Befehl ausfuhren

```
# Konfigurationsassistenten erneut aufrufen
openclaw configure
# Oder nur einen bestimmten Abschnitt konfigurieren
openclaw configure --section web
```

Sie konnen Konfigurationswerte auch direkt uber die CLI lesen und andern:

Methode A: API Key verwenden (fur alle Benutzer geeignet)

Bei Verwendung von Anthropic Claude (empfohlen, da OpenClaw nativ auf dem Claude-Okosystem basiert) benotigen Sie einen API Key von der Anthropic Console (https://console.anthropic.com/) (eine vollstandige Anleitung zur Multi-Modell-Schlusselkonfiguration finden Sie im «OpenClaw API Key Einrichtungsleitfaden»):

🔧 Befehl ausfuhren

```
# API Key uber CLI konfigurieren (unterstutzt Umgebungsvariablen-Ersetzung)
# In ~/.openclaw/openclaw.json kann geschrieben werden: "apiKey": "${ANTHROPIC_API_KEY}"
openclaw models auth paste-token --provider anthropic
```

Methode B: setup-token des Claude Code-Abonnements verwenden

Wenn Sie bereits ein Claude Code-Abonnement (Max / Pro Plan) haben, konnen Sie sich direkt mit dem setup-token authentifizieren, ohne zusatzlichen API Key:

🔧 Befehl ausfuhren

```
# Fuhren Sie zunachst in einem anderen Terminal claude setup-token aus, um den Token zu erhalten
openclaw models auth paste-token --provider anthropic
# Nach dem Einfugen des setup-token ist die Authentifizierung abgeschlossen
```

Standardmodell festlegen:

Befehl ausfuhren

```
# Aktuelles Standardmodell anzeigen
openclaw config get agents.defaults.model.primary
# Standardmodell festlegen (vereinfachter Befehl)
openclaw models set "anthropic/claude-opus-4-6"
# Fallback-Modell festlegen
openclaw config set agents.defaults.model.fallbacks '["openai/gpt-4o"]'
```

Bei Konfigurationsproblemen konnen Sie das eingebaute Diagnosetool verwenden:

Befehl ausfuhren

```
# Konfiguration uberprufen
openclaw doctor
# Haufige Probleme automatisch beheben
openclaw doctor --fix
```

> ✅ **Prufpunkt**
>
> Die Ausfuhrung von `openclaw doctor` sollte zeigen, dass alle Prufpunkte bestanden wurden. Nach korrekter Modellkonfiguration konnen Sie in der Control UI eine Testnachricht senden, um zu bestatigen, dass die KI normal antwortet.

STEP 3 Messaging-Kanale einrichten: Ihren Computer per Smartphone steuern ca. 5 Minuten

Dies ist eine der beeindruckendsten Funktionen von OpenClaw — uber WhatsApp, Telegram, Discord und andere Messaging-Apps Befehle an Ihren Computer senden. Der vierte Schritt des Onboarding-Assistenten umfasst bereits die Kanaleinrichtung, Sie konnen Kanale aber auch nachtraglich manuell hinzufugen.

OpenClaw verwendet einen **Pairing-Mechanismus** zur Verwaltung der Kanalzugriffsrechte. Bevor Sie jedoch ein Pairing durchfuhren, **muss zunachst die Vorkonfiguration des Kanals abgeschlossen werden**. Am Beispiel von Telegram:

Befehl ausfuhren

```
# 1. Telegram Bot Token konfigurieren (vom @BotFather erhalten)
openclaw config set channels.telegram.accounts.default.botToken "YOUR_BOT_TOKEN"
# 2. Telegram-Plugin aktivieren (standardmassig deaktiviert)
openclaw config set plugins.entries.telegram.enabled true
# 3. Gateway neu starten, damit die Konfiguration wirksam wird
openclaw daemon restart
```

Nach Abschluss der Vorkonfiguration kann das Pairing durchgefuhrt werden:

Befehl ausfuhren

```
# Gerate anzeigen, die auf Pairing warten
openclaw pairing list telegram
# Pairing genehmigen
openclaw pairing approve telegram <CODE>
```

Der WhatsApp-Kanal wird uber QR-Code-Scannen verbunden. Alle Kanaleinstellungen werden im Abschnitt `channels` von `~/.openclaw/openclaw.json` gespeichert. Die unterstutzten Zugriffssteuerungsrichtlinien umfassen:

- **pairing (Standard):** Neue Gerate mussen genehmigt werden, bevor sie mit OpenClaw kommunizieren konnen
- **allowlist:** Nur Benutzer auf der Whitelist sind zugelassen
- **open:** Fur alle offen (nicht empfohlen fur offentliche Szenarien)

Nach erfolgreicher Verbindung konnen Sie:

- Textbefehle von Ihrem Smartphone an OpenClaw senden, die auf dem Computer ausgefuhrt werden
- Ausfuhrungsergebnisse von OpenClaw als Ruckmeldung erhalten
- Ihren Computer unterwegs fernsteuern (vorausgesetzt, der Computer bleibt eingeschaltet und OpenClaw lauft weiter)

Derzeit unterstutzte Kanale: WhatsApp, Telegram, Discord, Slack, Signal, iMessage, Google Chat, Mattermost und MS Teams. Sie konnen auch uber die CLI schnell eine Testnachricht senden:

Befehl ausfuhren

```
# Testnachricht senden
openclaw message send --target +886912345678 --message "Hello from OpenClaw"
```

Senden Sie nach der Kanaleinrichtung eine Nachricht von Ihrem Smartphone an OpenClaw, z. B. „Hallo, sagen Sie mir bitte die aktuelle Uhrzeit". Wenn alles korrekt eingerichtet ist, antwortet OpenClaw innerhalb weniger Sekunden.

> ✅ **Prufpunkt**
>
> Senden Sie per Smartphone „Hallo" an OpenClaw — Sie sollten innerhalb weniger Sekunden eine Antwort erhalten. Die Ausfuhrung von `openclaw pairing list <channel>` sollte die verbundenen Gerate anzeigen.

STEP 4 Skills installieren und Hooks aktivieren

ca. 3 Minuten

Skills sind der Fahigkeitserweiterungsmechanismus von OpenClaw. Skills konnen auf **zwei Arten installiert** werden: automatische Erkennung von System-CLI-Tools und Installation uber den `clawhub` -Community-Paketmanager.

Methode A: System-CLI-Tools installieren (automatische Erkennung)

Befehl ausfuhren

```
# Browser-Automatisierung (benotigt Chromium)
sudo apt install -y chromium-browser
# GitHub-Integration (benotigt gh CLI + Authentifizierung)
sudo apt install -y gh
gh auth login --web
# Claude Code Integration
npm i -g @anthropic-ai/claude-code
# Gemini CLI
npm i -g @google/gemini-cli
# YouTube-Untertitel / Audio-Download
pip3 install yt-dlp
# Audio-Splitting + Videobearbeitung
sudo apt install -y ffmpeg ripgrep
```

Copy

Methode B: Community Skills uber clawhub installieren

`clawhub` ist der Community-Skills-Paketmanager von OpenClaw, mit dem Sie verschiedene Fahigkeitsmodule suchen und mit einem Klick installieren konnen:

Befehl ausfuhren

```
# Verfugbare Skills durchsuchen
npx clawhub search gemini
# Community Skill installieren (Installation in ~/.openclaw/workspace/skills/)
npx clawhub install sag          # ElevenLabs TTS
npx clawhub install nano-pdf     # PDF-Verarbeitung
npx clawhub install summarize    # Zusammenfassungstool
```

Copy

Methode C: API Key-Umgebungsvariablen konfigurieren

Einige Skills benotigen API Keys von Drittanbietern. Nach der Konfiguration mussen diese in die systemd-Dienstumgebung ubernommen werden (siehe Fallstricke #11):

Befehl ausfuhren

```
# OpenAI (Bildgenerierung, Sprache-zu-Text)
export OPENAI_API_KEY="Ihr_Key"
# Gemini
export GEMINI_API_KEY="Ihr_Key"
# ElevenLabs (Sprachsynthese)
export ELEVENLABS_API_KEY="Ihr_Key"
# Notion-Integration
export NOTION_API_KEY="Ihr_Key"
# Vergessen Sie nicht, diese auch in den systemd-Service und /home/coder/.profile einzutragen
```

Copy

Befehl ausfuhren

```
# Skills-Erkennungsstatus uberprufen
openclaw skills check
```

Copy

Methode D: Benutzerdefinierte Skills erstellen (SKILL.md)

Sie konnen haufig verwendete Workflows als benutzerdefinierte Skills verpacken und im Verzeichnis `~/.openclaw/workspace/skills/` ablegen. Jeder Skill benotigt lediglich eine `SKILL.md` -Datei, die OpenClaw automatisch erkennt und ladt:

Befehl ausfuhren

```
# Verzeichnis fur benutzerdefinierten Skill erstellen
mkdir -p ~/.openclaw/workspace/skills/my-skill
# SKILL.md Formatbeispiel (Frontmatter + Verwendungsanleitung):
cat > ~/.openclaw/workspace/skills/my-skill/SKILL.md << 'EOF'
---
name: my-skill
description: Einzeilige Beschreibung der Skill-Funktion
metadata: {"openclaw":{"emoji":"🔧","requires":{"bins":["curl"],"env":["MY_API_KEY"]}}}
---
# My Skill
Verwendungsanleitung, Befehlsbeispiele, Parameterbeschreibung...
EOF
# Uberprufen, ob der Skill erkannt wurde
openclaw skills check
```

In unseren Tests haben wir zwei benutzerdefinierte Skills erstellt:

- 🎬 **youtube-transcript** — Integriert `yt-dlp` -Untertiteldownload + OpenAI Whisper API-Audiotranskription, wahlt automatisch die beste Methode zur Erstellung des Transkripts
- 🎨 **gemini-image** — Generiert Bilder uber die Gemini 3 Pro Image Preview API, unterstutzt Stapelgenerierung und benutzerdefinierte Prompts

Praxis-Prompt-Beispiele fur Skills:

Im Folgenden finden Sie echte Befehlsbeispiele, die uber Telegram an OpenClaw gesendet wurden — jede Nachricht lost automatisch den entsprechenden Skill aus:

Skill	Telegram Prompt
🎬 youtube-transcript	Konvertiere dieses Video in ein deutsches Transkript https://youtube.com/watch?v=xxx
🎨 gemini-image	Generiere mit Gemini ein Bild einer Tokio-Nachtszene im Cyberpunk-Stil, dunkelblaue Tone mit goldenen Neonlichtern
🧩 coding-agent	Verwende Claude Code, um in ~/Projects/app eine OAuth Google-Anmeldefunktion hinzuzufugen
🐙 github	Prufe, ob bei hirosichen/my-repo aktuelle PRs ein Review benotigen
📝 notion	Organisiere die heutigen Besprechungsnotizen als Notion-Seite mit dem Titel „2026-02-14 Wochentreffen"
🖼️ openai-image-gen	Generiere mit DALL-E ein minimalistisches Produktschema auf weissem Hintergrund
☁️ openai-whisper-api	Konvertiere die Audiodatei /tmp/meeting.mp3 in Text
🗣️ sag	Konvertiere mit ElevenLabs den Text „Willkommen bei Meta Intelligence" in Sprache
♊ gemini	Analysiere mit Gemini die Datentrends in diesem Screenshot
🌤️ weather	Wie ist das aktuelle Wetter in Berlin

Die Hooks-Funktion (Webhooks) ermoglicht die automatische Ausfuhrung von Aktionen bei bestimmten Ereignissen und kann im Automatisierungsabschnitt von `~/.openclaw/openclaw.json` konfiguriert werden.

OpenClaw verfugt uber eine einzigartige Fahigkeit: **Es kann selbstständig Skills erstellen**. Wenn die KI feststellt, dass eine Aufgabe eine bestimmte Funktion erfordert und die vorhandenen Skills nicht ausreichen, kann sie selbstständig neue Fahigkeitsmodule schreiben und laden. Alle benutzerdefinierten Skills werden im Skills-Verzeichnis des Arbeitsbereichs gespeichert.

> ✅ **Prufpunkt**
>
> Die Ausfuhrung von `openclaw skills check` sollte eine Eligible-Anzahl grosser als 12 anzeigen. Damit ist die Basis-Bereitstellung abgeschlossen und Sie konnen mit den Praxisszenarien beginnen.

VIII. Praxisszenario 1 — Browser-Automatisierung

OpenClaw verfugt uber eine eingebaute Browser-Automatisierungsfunktion (`openclaw browser`), mit der sich **ein Chromium-Browser fernsteuern** lasst — Webseiten offnen, Informationen abrufen, Formulare ausfullen, Screenshots erstellen — alles automatisch.

Wenn Sie auf einem Cloud-Server (ohne grafische Oberflache) arbeiten, mussen Sie zunachst ein virtuelles Display installieren:

🔧 Befehl ausfuhren

```
# Cloud-Server (ohne grafische Oberflache) benotigt virtuelles Display
sudo apt install -y xvfb
Xvfb :99 -screen 0 1280x720x24 &
export DISPLAY=:99
```

Das Testen ist einfach — senden Sie einen Befehl uber WhatsApp oder das Terminal:

„Offne GitHub, suche nach dem OpenClaw-Projekt und teile mir die aktuelle Sternanzahl und die neueste Release-Version mit"

OpenClaw startet automatisch Chromium, navigiert zu GitHub, gibt den Suchbegriff ein, klickt auf die Projektseite, erfasst Sternanzahl und Release-Informationen und sendet Ihnen das Ergebnis als aufbereiteten Text zuruck.

Das Potenzial dieser Funktion geht weit uber die Suche hinaus — sie kann fur regelmasige Wettbewerberbeobachtung, automatisches Ausfullen von Formularen, periodische Webdatenextraktion und mehr eingesetzt werden. Beachten Sie jedoch, dass die Browser-Automatisierung mindestens 4 GB Arbeitsspeicher benotigt und auf Cloud-Servern eine zusatzliche Einrichtung des virtuellen Displays (Headless Chromium) erfordert.

IX. Praxisszenario 2 — Geplante Aufgaben: Taglicher KI-Bericht

Dies ist eine der nutzlichsten Funktionen in unseren Tests. Uber Cron-Job-Konfiguration kann OpenClaw taglich zu einer festgelegten Zeit automatisch Aufgaben ausfuhren und die Ergebnisse an Ihr WhatsApp senden.

In Kombination mit dem BlogWatcher-Skill konnen Sie einen taglichen KI-Nachrichtenbericht einrichten:

🔧 Befehl ausfuhren

```
# Zunachst mit BlogWatcher die zu uberwachenden RSS-Quellen hinzufugen
blogwatcher add "GitHub: Claude Code" \
  https://github.com/anthropics/claude-code/releases.atom
blogwatcher add "OpenAI Blog" \
  https://openai.com/blog/rss.xml
# Manuell einmal scannen, um die RSS-Quellen zu uberprufen
blogwatcher scan
blogwatcher articles
```

Nachdem Sie bestatigt haben, dass die RSS-Quellen funktionieren, richten Sie die geplante Aufgabe ein:

🔧 Befehl ausfuhren

```
# Jeden Morgen um 9 Uhr automatisch KI-Bericht an WhatsApp senden
openclaw cron add \
  --name "Taglicher KI-Bericht" \
  --cron "0 9 * * *" \
  --tz "Europe/Berlin" \
  --session isolated \
  --message "Bitte fassen Sie die wichtigsten KI-Nachrichten der letzten 24 Stunden auf Deutsch zusammen, einschliesslich technischer Durchbruche, Produktveroffentlichung
  --deliver \
  --channel whatsapp
```

Nach der Einrichtung erhalten Sie jeden Morgen um 9 Uhr einen automatisch von der KI zusammengestellten Nachrichtenbericht auf WhatsApp. Fur einen sofortigen Test:

🔧 Befehl ausfuhren

```
# Sofort ausfuhren (nicht auf die geplante Zeit warten)
openclaw cron run <CRON_JOB_ID> --force
```

Die Anwendungsmoglichkeiten fur geplante Aufgaben sind ausserst vielfaltig: tagliche Wettererinnerungen, automatische Wochenberichtserstellung, Wettbewerberbeobachtung, Social-Media-Datenauswertung und mehr — alles realisierbar mit einer einzigen Cron-Konfiguration.

X. Praxisszenario 3 — Claude Code fur automatische Entwicklung aufrufen

Dies ist die Funktion, die Technikbegeisterte am meisten begeistert — OpenClaw kann Claude Code aufrufen, um **automatisch Code zu schreiben, zu testen und bereitzustellen**.

In unseren Tests haben wir uber WhatsApp folgenden Befehl gesendet:

„Erstelle bitte mit Node.js + Express eine Backend-Anmeldeseite mit Benutzername- und Passwortfeldern, verwende Bootstrap fur das Design und verschlussle Passworter mit bcrypt"

Anschliessend hat OpenClaw Folgendes erledigt:

1. **Projektstruktur erstellt:** Automatisch Verzeichnisse angelegt und `package.json` initialisiert
2. **Abhangigkeiten installiert:** Automatisch `npm install express bcrypt ejs` ausgefuhrt
3. **Backend-Code geschrieben:** Express-Routen, bcrypt-Verschlusselungslogik und Session-Management generiert
4. **Frontend-Seite erstellt:** Eine Anmeldeseiten-Vorlage mit Bootstrap generiert
5. **Gestartet und getestet:** Den Server automatisch gestartet und die zugangliche URL zuruckgemeldet

Der gesamte Vorgang dauerte etwa 2–3 Minuten. Schliesslich offneten wir die zuruckgemeldete URL im Browser und sahen eine voll funktionsfahige Backend-Anmeldeseite.

Der Unterschied dieser Erfahrung „KI per naturlicher Sprache zum Programmieren anweisen" gegenuber der direkten Nutzung von Claude oder ChatGPT liegt darin: **OpenClaw generiert nicht nur Code-Snippets, sondern erledigt den gesamten Prozess von der Projekterstellung bis zur Bereitstellung**. Es ubernimmt eigenstandig Dateisystemoperationen, Paketinstallationen und Umgebungskonfigurationen — all diese Schritte, die Entwickler normalerweise manuell erledigen mussen[5].

XI. Fortgeschritten: Hooks Zero-Polling + Agent Teams Multi-Agenten-Zusammenarbeit

Die im vorherigen Abschnitt beschriebene konventionelle Aufrufmethode hat einen Schmerzpunkt: **OpenClaw pollt alle paar Sekunden den Status und die Ausgabe von Claude Code**. Je langer die Aufgabenausfuhrung dauert, desto mehr Polling-Zyklen und desto mehr Token-Verbrauch. Dies ist das in der Community am haufigsten genannte Problem.

Die Losung ist uberraschend elegant — nutzen Sie den **Hooks-Callback-Mechanismus** von Claude Code in Kombination mit dem neuesten Feature **Agent Teams Multi-Agenten-Zusammenarbeit**, um echtes Zero-Polling fur asynchrone Entwicklung zu erreichen.

11.1 Kernprinzip: Vom Polling zum Callback

Der traditionelle Ablauf ist:

OpenClaw erteilt Aufgabe → Pollt alle paar Sekunden den Claude Code-Status → Verbraucht laufend Token → Aufgabe abgeschlossen → Ergebnis zuruckgeben

Die Hooks Zero-Polling-Methode funktioniert so:

OpenClaw delegiert Aufgabe (einmalig, im Hintergrund) → Claude Code lauft eigenstandig → Aufgabe abgeschlossen, Stop Hook wird ausgelost → Ergebnis automatisch geschrieben + OpenClaw geweckt → Benachrichtigung an Chat-Gruppe gepusht

Wahrend des gesamten Prozesses verbraucht OpenClaw nur beim **Erteilen der Aufgabe** und beim **Lesen des Ergebnisses** jeweils Token. Wahrend Claude Code eigenstandig entwickelt, werden keine OpenClaw-Token verbraucht. Ausserdem wird der Haupt-Agent nicht blockiert und kann gleichzeitig andere Aufgaben ausfuhren.

11.2 Agent Teams: Multi-Agenten-Zusammenarbeit von Claude Code

Die kurzlich hinzugefugte Agent Teams-Funktion von Claude Code macht diesen Workflow noch leistungsfahiger. Agent Teams entsprechen einem vollstandigen Entwicklungsteam innerhalb von Claude Code — jeder Agent ist ein eigenstandiger Prozess, der wirklich parallel ausgefuhrt wird. Die Agenten konnen miteinander kommunizieren, Aufgabenlisten teilen, Arbeit automatisch ubernehmen und spezialisierte Rollen (Frontend, Backend, Testing usw.) einnehmen.

In Kombination mit Hooks-Callbacks konnen Sie uber Ihr Smartphone einen Entwicklungsbefehl an OpenClaw senden. Die Agent Teams von Claude Code teilen die Arbeit automatisch auf, mehrere Agenten arbeiten zusammen, und nach Fertigstellung wird automatisch ein detaillierter Bericht an Ihre Chat-Gruppe gepusht.

11.3 Hooks-Konfiguration: Stop + SessionEnd — doppelte Absicherung

Von den 14 Hooks in Claude Code verwenden wir zwei:

- **Stop Hook (Haupt-Callback):** Wird ausgelost, wenn Claude Code die Generierung abgeschlossen hat, und stellt sicher, dass der Callback erst nach echter Fertigstellung der Entwicklung erfolgt
- **SessionEnd Hook (Absicherungs-Callback):** Wird beim Sitzungsende ausgelost und dient als Sicherheitsbackup — selbst wenn der Stop Hook fehlschlagt, wird SessionEnd trotzdem eine Benachrichtigung auslosen

Voraussetzung: Claude Code CLI-Authentifizierung (drei Ebenen)

Der `coding-agent` -Skill von OpenClaw fuhrt die Claude Code CLI als Nicht-Root-Benutzer `coder` aus (Sicherheitsisolation). Das bedeutet, dass Sie die Authentifizierung an **drei Stellen** konfigurieren mussen: Root-Shell, systemd-Service und coder-Benutzer.

🔧 Befehl ausfuhren — Step A: Token erhalten

```
# OAuth Token auf dem lokalen Computer erhalten (ein Jahr gultig)
claude setup-token
```

Copy

🔧 Befehl ausfuhren — Step B: coder-Benutzer erstellen

```
# coder-Benutzer erstellen (OpenClaw wechselt automatisch zu diesem Benutzer)
useradd -m -s /bin/bash -U coder
# Passwortloses su erlauben (von OpenClaw intern benotigt)
passwd -d coder
```

Copy

🔧 Befehl ausfuhren — Step C: Drei-Ebenen-Authentifizierung

```
# Ebene 1: Root-Shell (fur direkte SSH-Nutzung)
echo 'export CLAUDE_CODE_OAUTH_TOKEN="Ihr_Token"' >> ~/.bashrc
# Ebene 2: systemd-Service (OpenClaw Gateway vererbt an Kindprozesse)
# Bearbeiten Sie ~/.config/systemd/user/openclaw-gateway.service
# Fugen Sie im [Service]-Abschnitt hinzu:
#   Environment=CLAUDE_CODE_OAUTH_TOKEN=Ihr_Token
systemctl --user daemon-reload && openclaw daemon restart
# Ebene 3: coder-Benutzer (tatsachliche Ausfuhrungsumgebung des coding-agent)
echo 'export CLAUDE_CODE_OAUTH_TOKEN="Ihr_Token"' >> /home/coder/.profile
chown coder:coder /home/coder/.profile
```

Copy

🔧 Befehl ausfuhren — Step D: Verifizierung

```
# Root-Umgebung verifizieren
claude -p "hello"
# coder-Umgebung verifizieren
su - coder -s /bin/bash -c "claude -p 'hello'"
```

Copy

Hinweis: Die Hook-Skripte mussen ebenfalls in das Verzeichnis des coder-Benutzers (`/home/coder/.openclaw/hooks/`) kopiert werden, und die Pfade in `/home/coder/.claude/settings.json` mussen aktualisiert werden. Da der coder-Benutzer keinen Zugriff auf die Root-OpenClaw-Konfiguration hat, sollten Telegram-Benachrichtigungen im Hook uber `curl` direkt an die Bot-API gesendet werden, anstatt `openclaw message send` zu verwenden.

Konfigurieren Sie die Hooks in der Konfigurationsdatei `~/.claude/settings.json` von Claude Code:

📄 Konfigurationsdatei — ~/.claude/settings.json

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/hooks/notify-agi.sh",
            "timeout": 10
          }
        ]
      }
    ],
    "SessionEnd": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "/path/to/hooks/notify-agi.sh",
            "timeout": 10
          }
        ]
      }
    ]
  }
}
```

Hinweis: Claude Code ubergibt Kontextinformationen im JSON-Format uber **stdin** (einschliesslich `session_id` , `cwd` , `hook_event_name`). Das Callback-Skript muss diese Daten aus stdin lesen. Da Stop und SessionEnd kurz nacheinander ausgelost werden konnen, sollte das Skript einen **Deduplizierungsmechanismus** (z. B. eine Lock-Datei) implementieren, um doppelte Benachrichtigungen zu vermeiden.

11.4 Zweikanal-Design des Callback-Skripts

Das Callback-Skript verwendet eine **Zweikanal**-Architektur, die Zuverlassigkeit und Echtzeit-Fahigkeit vereint:

- **Data Channel (latest.json):** Schreibt das vollstandige Aufgabenergebnis (Session-ID, Zeitstempel, Arbeitsverzeichnis, vollstandige Ausgabe, Status) in eine JSON-Datei ohne Langenbeschrankung
- **Signal Channel (Wake Event):** Weckt die Hauptsitzung von OpenClaw uber einen API-Aufruf und liefert Echtzeitbenachrichtigungen

Der Grund fur zwei Kanale ist, dass Wake Events eine Zeichenbegrenzung haben (ca. 300 Zeichen), wahrend die Ausgabe von Claude Code uber 2000 Zeichen betragen kann. Die Dateispeicherung nimmt unbegrenzte Inhalte auf, das Wake Event ist fur die Echtzeitbenachrichtigung zustandig. Selbst wenn der Gateway-API-Aufruf fehlschlagt, wird latest.json trotzdem geschrieben, und der Agent kann das Ergebnis beim nachsten Heartbeat lesen — das ist das fehlertolerante Design.

Das Wecken von OpenClaw erfolgt uber die CLI durch Injektion eines Systemereignisses (Hinweis: Das Gateway bietet keine REST-API; es muss die CLI verwendet werden):

📖 Nur als Referenz

```
# Methode 1: CLI-direkte Systemereignis-Injektion (empfohlen)
openclaw system event \
  --text "Claude Code task complete, read latest.json" \
  --mode now \
  --token "$TOKEN"
# Methode 2: Telegram-Nachricht direkt an die angegebene Gruppe senden
openclaw message send \
  --channel telegram \
  --target "$TELEGRAM_GROUP" \
  --message "🤖 Claude Code Aufgabe abgeschlossen, Ergebnis in latest.json geschrieben"
```

11.5 Praxis: Telegram-Benachrichtigungsgruppe einrichten

Damit der Hook nach Abschluss automatisch Benachrichtigungen an Telegram sendet, benotigen Sie eine **separate Benachrichtigungsgruppe** (getrennt vom Hauptfenster fur tagliche Gesprache, um Kontextverwirrung zu vermeiden). Hier sind die vollstandigen Einrichtungsschritte:

Erster Schritt: Gruppe erstellen und Gruppen-ID ermitteln

1. Erstellen Sie eine neue Gruppe in Telegram (z. B. „OpenClaw Benachrichtigungen")
2. Fugen Sie Ihren OpenClaw Bot zur Gruppe hinzu
3. Senden Sie eine beliebige Nachricht in der Gruppe

Beim Ermitteln der Gruppen-ID gibt es eine Tucke: Wenn das OpenClaw Gateway lauft, konsumiert es fortlaufend die Telegram-Updates, wodurch die `getUpdates` -API ein leeres Array zuruckgibt. Die Losung ist, die **Gateway-Logs zu uberprufen**:

🔧 Befehl ausfuhren

```
# Gruppen-ID aus den Gateway-Logs ermitteln
tail -200 /tmp/openclaw/openclaw-*.log | grep -o '"chatId":-[0-9]*'
# Beispielausgabe: "chatId":-5201877902
```

Zweiter Schritt: Gruppenzugriffsrechte konfigurieren

Die Standard- `groupPolicy: allowlist` blockiert alle Gruppennachrichten. Andern Sie sie auf `open` (oder fugen Sie die Gruppen-ID zur Whitelist hinzu):

🔧 Befehl ausfuhren

```
# Gruppenzugriff offnen (oder je nach Bedarf allowlist konfigurieren)
openclaw config set channels.telegram.groupPolicy open
openclaw config set channels.telegram.accounts.default.groupPolicy open
# Neu starten, damit die Konfiguration wirksam wird
openclaw daemon restart
```

Copy

Dritter Schritt: Benachrichtigungsversand testen

🔧 Befehl ausfuhren

```
# Mit Ihrer Gruppen-ID testen (durch die tatsachliche ID ersetzen)
openclaw message send \
  --channel telegram \
  --target "-5201877902" \
  --message "✅ OpenClaw Benachrichtigungstest erfolgreich"
```

Copy

Nachdem Sie bestatigt haben, dass die Gruppe die Nachricht erhalten hat, tragen Sie die Gruppen-ID in die Variable `TELEGRAM_GROUP` des Callback-Skripts ein. Danach wird der Hook bei Abschluss jeder Claude Code-Aufgabe automatisch eine Ergebniszusammenfassung an diese Benachrichtigungsgruppe senden.

11.6 Vollstandiger Ablauf: Vom Smartphone-Befehl bis zum Erhalt des Entwicklungsberichts

Im Folgenden ein vollstandiges Praxisszenario. Wir haben uber die Chat-App eingegeben:

„Verwende den Agent Teams-Kollaborationsmodus von Claude Code, um ein physikbasiertes HTML/CSS-Sandsimulationsspiel mit Materialsystem zu erstellen"

Was dann passierte:

1. OpenClaw delegierte die Aufgabe an die Agent Teams von Claude Code (**nur dieser eine Aufruf, Token-Verbrauch vernachlassigbar**)
2. Der Haupt-Agent wurde nicht blockiert — wir fragten im selben Chat-Fenster weiter „Wie ist das Wetter heute in Singapur?" und „Erzahl einen Witz", und OpenClaw antwortete sofort
3. Claude Code lief im Hintergrund etwa 6 Minuten lang, mehrere Agenten arbeiteten parallel zusammen
4. Die Entwicklung war abgeschlossen, der Stop Hook wurde automatisch ausgelost, das Ergebnis in latest.json geschrieben
5. Wir erhielten in einer **separaten Chat-Gruppe** eine Push-Benachrichtigung — mit Aufgabenname, Projektpfad, Abschlusszeit, Agent Teams-Aktivierungsstatus, 184 bestandenen Tests, Liste der gelieferten Funktionen und Projektstruktur

Der Grund fur die Benachrichtigung in einer separaten Gruppe statt im Hauptchat-Fenster des Agenten ist, dass eine plotzliche Abschlussbenachrichtigung wahrend der Ausfuhrung anderer Aufgaben im Hauptfenster zu Kontextverwirrung fuhren wurde.

Der vollstandige Hooks-Callback-Code ist Open Source verfugbar unter github.com/win4r/claude-code-hooks (https://github.com/win4r/claude-code-hooks), einschliesslich Dispatch-Skript, Callback-Skript und Claude Code-Konfigurationsbeispielen.

XII. Fortgeschritten: Supermemory-Plugin — Perfektes Langzeitgedachtnis fur KI

Die eingebaute Memory-Schicht von OpenClaw speichert Gesprachskontexte in Markdown-Dateien, doch mit zunehmender Nutzungsdauer stossen diese einfachen Gedachtnismechanismen an ihre Grenzen: begrenztes Kontextfenster, keine kanalubergreifende einheitliche Erinnerung, fehlende semantische Suchfahigkeit. Das Supermemory (https://github.com/supermemoryai/openclaw-supermemory)-Plugin wurde genau zur Losung dieser Probleme entwickelt.

12.1 Welches Problem lost Supermemory

Stellen Sie sich folgendes Szenario vor: Sie haben OpenClaw letzte Woche uber WhatsApp mitgeteilt „Ich bevorzuge die Entwicklung mit TypeScript", und heute bitten Sie es uber Telegram, eine API zu schreiben. **Ohne Supermemory** wurde OpenClaw in der Telegram-Sitzung nichts von Ihrer Praferenz wissen — da es sich um einen separaten Gesprachskontext handelt.

Mit installiertem Supermemory verfugt OpenClaw uber ein kanalubergreifendes, zeitubergreifendes, einheitliches semantisches Gedachtnis. Es wird:

- **Auto-Capture:** Nach jeder Gesprachsrunde automatisch Schlusselinformationen extrahieren (Ihre Praferenzen, Projekthintergrund, Entscheidungsprotokolle) und an die Supermemory-Cloud zur Deduplizierung und semantischen Indexierung senden
- **Auto-Recall:** Vor jeder KI-Antwort automatisch semantisch relevante historische Erinnerungen abfragen und in den Kontext injizieren — nicht alle Gesprache pauschal, sondern prazise die relevantesten Fragmente
- **User Profile:** Automatisch Ihr personliches Praferenzprofil erstellen und kontinuierlich aktualisieren — bevorzugte Sprachen, Technologie-Stack, Arbeitsgewohnheiten, Kommunikationsstil

Das Ergebnis: OpenClaw wird wirklich zu einem langfristigen Assistenten, der „Sie mit zunehmender Nutzung immer besser versteht", und nicht nur ein Gesprachstool, das jedes Mal bei null beginnt.

12.2 Installation und Konfiguration

Die Installation erfordert nur einen Befehl:

🔧 Befehl ausfuhren

```
openclaw plugins install @supermemory/openclaw-supermemory
```

Starten Sie OpenClaw nach der Installation neu. Konfigurieren Sie dann den API Key (erfordert Supermemory Pro (https://console.supermemory.ai) oder hoher):

🔧 Befehl ausfuhren

```
# Umgebungsvariable konfigurieren
export SUPERMEMORY_OPENCLAW_API_KEY="sm_your_key_here"
```

Oder direkt in `~/.openclaw/openclaw.json` eintragen:

📄 Konfigurationsdatei — ~/.openclaw/openclaw.json

```
{
  "plugins": {
    "entries": {
      "openclaw-supermemory": {
        "enabled": true,
        "config": {
          "apiKey": "${SUPERMEMORY_OPENCLAW_API_KEY}"
        }
      }
    }
  }
}
```

12.3 Erweiterte Konfigurationsparameter

Supermemory bietet feingranulare Steuerungsoptionen:

- **`containerTag` :** Gedachtnis-Namespace (Standard: `openclaw_{hostname}`), mehrere Rechner konnen sich denselben Gedachtnisspeicher teilen
- **`autoRecall` / `autoCapture` :** Schalter fur automatischen Abruf und automatische Erfassung (beide standardmassig `true`)
- **`maxRecallResults` :** Maximale Anzahl der pro Abfrage injizierten Erinnerungen (Standard: 10)
- **`profileFrequency` :** Alle wie viele Runden das vollstandige Benutzerprofil injiziert wird (Standard: 50)

12.4 Verwendung

Die meisten Funktionen laufen nach der Installation automatisch, Sie konnen aber auch manuell eingreifen:

🔧 Befehl ausfuhren

```
# Bestimmte Informationen manuell merken
/remember Mein Unternehmen verwendet AWS als Cloud-Infrastruktur und bevorzugt Terraform zur Verwaltung
# Gedachtnis aktiv durchsuchen
/recall Das letzte Mal besprochene Datenbankmigrationsstrategie
# CLI-Gedachtnisoperationen
openclaw memory search "API-Design-Praferenzen"
openclaw memory status              # Gedachtnis-Indexstatus anzeigen
openclaw memory index               # Gedachtnisindex neu aufbauen
```

Die KI kann auch selbststandig Gedachtniswerkzeuge aufrufen: `supermemory_store` (speichern), `supermemory_search` (suchen), `supermemory_forget` (loschen), `supermemory_profile` (Benutzerprofil lesen).

12.5 Anwendungsszenarien

Die Kombination von Supermemory und OpenClaw ist in folgenden Szenarien besonders wertvoll:

- **Langfristiges Projektmanagement:** Bei Entwicklungsprojekten uber mehrere Wochen erinnert sich OpenClaw an alle Architekturentscheidungen, Grunde fur Technologieauswahl und offene Aufgaben — Sie mussen nicht jedes Mal den Hintergrund neu erklaren
- **Nahtloser Geratewechsel:** Eine morgens am Computer in der Control UI besprochene Strategie kann nachmittags auf dem Smartphone uber WhatsApp direkt fortgesetzt werden — das Gedachtnis ist kanalubergreifend vereinheitlicht
- **Gemeinsame Team-Wissensbasis:** Uber die `containerTag` -Konfiguration fur einen gemeinsamen Namespace konnen mehrere Personen denselben OpenClaw-Gedachtnisspeicher nutzen und kollektives Teamwissen aufbauen
- **Personalisierte Automatisierung:** Geplante Aufgaben kombiniert mit Gedachtnis — beispielsweise passt der tagliche KI-Bericht die Inhaltsprioritaten automatisch an Ihre bisherigen Lesepraferenzen an
- **Kundenservice-Szenario:** OpenClaw erinnert sich an die Interaktionshistorie, Praferenzen und Problemprotokolle jedes Kunden und bietet ein wirklich personalisiertes Serviceerlebnis

XIII. Ubersicht der Kernkonfigurationsdateien

Wahrend der Nutzung mussen Sie moglicherweise einige Einstellungen manuell anpassen. Hier sind die wichtigsten Dateispeicherorte von OpenClaw:

- **`~/.openclaw/openclaw.json`:** Hauptkonfigurationsdatei (JSON5-Format), enthalt alle Einstellungen fur Modelle, Kanale, Agenten, Automatisierung usw. Das Gateway erkennt Dateiandernungen automatisch und ladt die Konfiguration dynamisch neu
- **`~/.openclaw/.env`:** Umgebungsvariablen-Datei, geeignet zur Speicherung sensibler Informationen wie API Keys
- **`~/.openclaw/workspace`:** Standard-Arbeitsbereich, in dem von OpenClaw generierte Dateien abgelegt werden

Sie konnen die Standardpfade auch uber Umgebungsvariablen uberschreiben:

- `OPENCLAW_HOME` — Pfad zum Hauptverzeichnis festlegen
- `OPENCLAW_STATE_DIR` — Statusverzeichnis-Speicherort uberschreiben
- `OPENCLAW_CONFIG_PATH` — Konfigurationsdateipfad uberschreiben

Haufig verwendete CLI-Konfigurationsbefehle:

🔧 Befehl ausfuhren

```
# Bestimmten Konfigurationswert lesen
openclaw config get agents.defaults.workspace
# Konfigurationswert andern
openclaw config set agents.defaults.heartbeat.every "2h"
# Konfigurationswert entfernen
openclaw config unset tools.web.search.apiKey
# Diagnose und automatische Reparatur
openclaw doctor
openclaw doctor --fix
```

Copy

Wenn OpenClaw ein unerwartetes Verhalten zeigt, fuhren Sie zunachst `openclaw doctor` zur Diagnose aus. Fur ein vollstandiges Zurucksetzen konnen Sie das Verzeichnis `~/.openclaw/` loschen und dann `openclaw setup` erneut ausfuhren.

> ### ⚠️ Praxiserfahrungen und Fallstricke: Zwolf haufige Probleme bei der Cloud-Server-Bereitstellung
>
> Im Folgenden finden Sie die Fallstricke, auf die wir bei der tatsachlichen Bereitstellung von OpenClaw v2026.2.12 auf einem Vultr VPS (Ubuntu 22.04) gestossen sind. Diese Probleme werden in der offiziellen Dokumentation nicht ausreichend erlautert, konnen Ihre Bereitstellung jedoch um Dutzende von Minuten verzogern oder sogar zum Scheitern bringen.
>
> 1. **Der Onboarding-Assistent schlagt in SSH-Umgebungen direkt fehl.** `openclaw onboard --install-daemon` benotigt ein interaktives TTY-Terminal und gibt in Remote-SSH-, Docker-Container- oder CI/CD-Umgebungen direkt einen Fehler aus. Losung: Verwenden Sie `openclaw setup` + `openclaw daemon install` fur die manuelle Ersteinrichtung
> 2. **Das Telegram-Plugin ist standardmassig deaktiviert.** Selbst wenn Sie den Bot Token konfiguriert haben, zeigt `openclaw pairing list telegram` keine Reaktion — weil das Telegram-Plugin standardmassig auf `enabled: false` steht. Sie mussen manuell `openclaw config set plugins.entries.telegram.enabled true` ausfuhren und den Daemon neu starten
> 3. **Der Befehl `openclaw skills install` existiert nicht.** Skills werden auf zwei Arten installiert: (1) Automatische Erkennung von System-CLI-Tools (z. B. `gh`, `chromium-browser`), (2) Installation von Community Skills uber `npx clawhub install <name>` (z. B. `sag`, `nano-pdf`). Einige Skills benotigen zusatzlich API Key-Umgebungsvariablen (z. B. `OPENAI_API_KEY`, `GEMINI_API_KEY`)
> 4. **Browser-Automatisierung kann auf Headless-Servern nicht gestartet werden.** Chromium benotigt einen Display-Server. Cloud-Server mussen `xvfb` installieren und `DISPLAY=:99` setzen, andernfalls erhalten Sie einen `cannot open display`-Fehler
> 5. **Das Gateway bietet keine REST-API zum Wecken des Agenten.** Der im Internet kursierende `curl -X POST /api/cron/wake` liefert 405 Method Not Allowed. Die korrekte Methode ist die CLI: `openclaw system event --text "..." --mode now --token "$TOKEN"`
> 6. **Die Kontextdaten der Claude Code Hooks werden als JSON uber stdin ubergeben.** Nicht als Umgebungsvariablen. Das Callback-Skript muss `session_id`, `cwd`, `hook_event_name` und andere Felder aus stdin lesen. Das Ignorieren von stdin fuhrt dazu, dass keine Aufgabeninformationen abgerufen werden konnen
> 7. **Stop- und SessionEnd-Hook werden kurz nacheinander ausgelost.** Ohne Deduplizierung (z. B. 30-Sekunden-Lock-Datei) erhalten Sie nach Abschluss derselben Aufgabe zwei Benachrichtigungen. Referenzimplementierung unter claude-code-hooks (https://github.com/win4r/claude-code-hooks)
> 8. **Der Modellkonfigurationsbefehl hat eine vereinfachte Version.** Sie mussen nicht den vollstandigen Pfad `openclaw config set agents.defaults.model.primary "anthropic/..."` schreiben, sondern konnen direkt `openclaw models set "anthropic/claude-opus-4-6"` verwenden. Claude Code-Abonnenten konnen zudem `openclaw models auth paste-token` fur die Authentifizierung ohne API Key nutzen
> 9. **Claude Code CLI und OpenClaw haben getrennte Authentifizierungen.** Selbst wenn OpenClaw uber `paste-token` authentifiziert ist, zeigt die Claude Code CLI weiterhin „Not logged in". Das Format von `.credentials.json` ist extrem schwer manuell zu erstellen. Die korrekte Methode: Fuhren Sie lokal `claude setup-token` aus, um den OAuth-Token zu erhalten, und setzen Sie dann auf dem Remote-Server `export CLAUDE_CODE_OAUTH_TOKEN="token"` als Umgebungsvariable
> 10. **coding-agent fuhrt Claude Code als `coder`-Benutzer aus, nicht als root.** OpenClaw wechselt intern zu `su coder`, um die Claude Code CLI auszufuhren. Wenn der `coder`-Benutzer nicht existiert, das Passwort nicht geloscht wurde oder dieser Benutzer kein `CLAUDE_CODE_OAUTH_TOKEN` hat, erhalten Sie Exit Code 2 oder „Authentication failure". Sie mussen manuell `useradd -m -s /bin/bash -U coder && passwd -d coder` ausfuhren und den Token in `/home/coder/.profile` konfigurieren
> 11. **Der systemd-Service liest `.bashrc` nicht.** Selbst wenn Sie `CLAUDE_CODE_OAUTH_TOKEN` in der `.bashrc` von root konfiguriert haben, werden die Kindprozesse des OpenClaw Gateways (systemd-verwaltet) dies nicht erben. Sie mussen im Abschnitt `[Service]` von `~/.config/systemd/user/openclaw-gateway.service` die Zeile `Environment=CLAUDE_CODE_OAUTH_TOKEN=token` hinzufugen und dann `systemctl --user daemon-reload` ausfuhren

12. **`npx clawhub install` installiert nur die Skill-Definition, nicht die CLI-Binary.** Beispielsweise erstellt `npx clawhub install sag` eine `SKILL.md` in `~/.openclaw/workspace/skills/sag/` (damit OpenClaw weiss, wie dieser Skill verwendet wird), installiert aber nicht die tatsachliche `sag`-Ausfuhrungsdatei. Sie mussen die Binary selbst installieren — fur sag unter Linux von GitHub Releases (https://github.com/steipete/sag/releases) herunterladen: `curl -sL .../sag_linux_amd64.tar.gz | tar xz && mv sag /usr/local/bin/`

XIV. Sicherheitshinweise: Nicht zu ignorierende Risiken

Nach der erfolgreichen Bereitstellung von OpenClaw mussen wir ernsthaft auf einige **schwerwiegende Sicherheitsrisiken** hinweisen.

Anfang Februar 2026 haben Sicherheitsforscher die Schwachstelle CVE-2026-25253 (CVSS 8.8 hochkritisch) offengelegt[6] — Angreifer konnen uber Cross-Site-WebSocket-Hijacking die vollstandige Kontrolle uber OpenClaw erlangen. CrowdStrikes Untersuchung ergab, dass von uber 42.000 offentlich zuganglichen OpenClaw-Instanzen im Internet 93,4 % eine Authentifizierungsumgehungsschwachstelle aufweisen[7].

Das Sicherheitsteam von Cisco stellte klar fest[8], dass personliche KI-Agenten wie OpenClaw mit ihrer Kombination aus Shell-Zugriff, Netzwerkverbindung und Prompt-Injection-Angriffsflache ideale Ziele fur Hacker darstellen.

Unsere Sicherheitsempfehlungen:

- **Setzen Sie OpenClaw niemals dem offentlichen Internet aus.** Fur den Fernzugriff empfiehlt die offizielle Dokumentation SSH Tunnel oder Tailscale — OpenClaw bietet eine eingebaute Tailscale-Integration, die direkt wahrend des Onboarding konfiguriert werden kann
- **Aktivieren Sie den Docker-Sandbox-Modus.** OpenClaw unterstutzt Docker-isolierte Ausfuhrung (Sandboxing), die fur Nicht-Hauptsitzungen oder alle Sitzungen aktiviert werden kann, um zu verhindern, dass nicht vertrauenswurdige Eingaben direkt auf das System zugreifen
- **Aktualisieren Sie regelmassig.** Sicherheitspatches fur OpenClaw werden haufig veroffentlicht — halten Sie stets die neueste Version bereit
- **Beschranken Sie API Key-Berechtigungen.** Konfigurieren Sie fur die von OpenClaw verwendeten API Keys minimale Berechtigungen und Verbrauchslimits
- **Verwenden Sie es nicht in Produktionsumgebungen.** Im aktuellen Stadium eignet sich OpenClaw fur technische Exploration und Proof of Concept, jedoch noch nicht fur die Verarbeitung sensibler Geschaftsdaten

Quick Reference — Befehlsubersicht

Befehl	Verwendungszweck
`openclaw setup && openclaw daemon install`	Ersteinrichtung und Daemon-Service-Installation
`openclaw gateway status`	Gateway-Betriebsstatus uberprufen
`openclaw dashboard`	Web-basierte Steuerungsoberflache offnen
`openclaw configure`	Konfigurationsassistenten erneut aufrufen
`openclaw models set <model>`	Standardmodell festlegen
`openclaw config get/set <key>`	Einzelne Konfigurationswerte lesen oder andern
`openclaw doctor --fix`	Probleme diagnostizieren und automatisch beheben
`openclaw pairing list <channel>`	Kanal-Pairing-Status anzeigen
`openclaw skills check`	Erkannte Skills-Status uberprufen
`npx clawhub search/install <name>`	Community Skills suchen oder installieren
`openclaw cron add ...`	Neue geplante Aufgabe hinzufugen
`openclaw cron run <id> --force`	Geplante Aufgabe sofort ausfuhren

XV. Docker Schnellbereitstellung — Vollstandige Umgebung mit einem Klick replizieren

Nach dem Durcharbeiten der vorherigen vierzehn Kapitel denken Sie moglicherweise: Die Einrichtungsschritte sind zwar klar, aber **die Anzahl der manuell zu installierenden Abhangigkeiten ist wirklich zu gross** — Node.js, Chromium, yt-dlp, sag, ffmpeg, ripgrep, GitHub CLI, dazu die dreistufige Claude Code-Authentifizierung, Telegram-Plugin-Konfiguration, benutzerdefinierte Skills... bei jedem Schritt konnen Versionsinkompatibilitaten oder Umgebungsunterschiede auftreten.

Docker lost genau dieses Problem. Wir haben die gesamte OpenClaw-Bereitstellungsumgebung in ein Docker-Image verpackt — **ein einziger Befehl stellt die vollstandige Arbeitsumgebung wieder her** — einschliesslich aller Systemabhangigkeiten, globalen NPM-Pakete, CLI-Tools, benutzerdefinierten Skills und Hooks-Callback-Skripte. Fur eine vollstandigere Losung mit Docker Compose Multi-Container-Orchestrierung, Nginx-Reverse-Proxy, systemd-Dienstverwaltung und Cloud-Bereitstellung lesen Sie bitte den «OpenClaw Docker Bereitstellungsleitfaden».

15.1 Image-Inhalt

Dieses Docker-Image basiert auf `node:22-bookworm` (Debian 12 + Node.js 22) und enthalt folgende vorinstallierte Komponenten:

- **Systemtools:** python3, jq, curl, wget, git, xvfb, chromium, ripgrep, ffmpeg
- **Globale NPM-Pakete:** openclaw, @anthropic-ai/claude-code, @google/gemini-cli

- **CLI-Binaries:** gh (GitHub CLI), yt-dlp, sag (ElevenLabs TTS)
- **Benutzerdefinierte Skills:** youtube-transcript (Untertitelextraktion), gemini-image (Bildgenerierung)
- **Hooks-Skripte:** Stop + SessionEnd-Doppel-Callback mit Telegram-Benachrichtigungsunterstutzung
- **coder-Benutzer:** Bereits erstellt und fur passwortloses su konfiguriert (benotigt vom OpenClaw coding-agent)

15.2 Image erstellen

🔧 Befehl ausfuhren

```
git clone https://github.com/hirosichen/metaintelligence-2026.git
cd metaintelligence-2026
docker build -t openclaw-stack docker/openclaw/
```

Der Build-Prozess dauert etwa 3–5 Minuten (abhangig von der Netzwerkgeschwindigkeit), das endgultige Image ist etwa 2–3 GB gross.

15.3 Container starten

Beim Start werden API Keys und Authentifizierungsinformationen uber Umgebungsvariablen ubergeben — **diese sensiblen Daten werden nicht in das Image eingebrannt**:

🔧 Befehl ausfuhren

```
docker run -d --name openclaw \
  -e CLAUDE_CODE_OAUTH_TOKEN="your-oauth-token" \
  -e ANTHROPIC_API_KEY="sk-ant-..." \
  -e TELEGRAM_BOT_TOKEN="123456:ABC..." \
  -e TELEGRAM_GROUP_ID="-5201877902" \
  -e OPENAI_API_KEY="sk-..." \
  -e GEMINI_API_KEY="AIza..." \
  -e ELEVENLABS_API_KEY="..." \
  -e OPENCLAW_MODEL="anthropic/claude-opus-4-6" \
  -p 18789:18789 \
  openclaw-stack
```

Dabei sind nur `CLAUDE_CODE_OAUTH_TOKEN` und `ANTHROPIC_API_KEY` erforderlich, die ubrigen sind optional. `CLAUDE_CODE_OAUTH_TOKEN` kann lokal uber `claude setup-token` erhalten werden.

15.4 Bereitstellung verifizieren

🔧 Befehl ausfuhren

```
# Container-Logs uberprufen
docker logs openclaw
# In den Container eintreten
docker exec -it openclaw bash
# Gateway-Status uberprufen
docker exec openclaw openclaw gateway status
# Skills-Erkennung uberprufen
docker exec openclaw openclaw skills check
# Claude Code testen (als coder-Benutzer)
docker exec openclaw su - coder -s /bin/bash -c "claude -p 'hello'"
```

15.5 Hinweise

- **API Keys sind personlich:** Das Image enthalt keine Authentifizierungsinformationen. Jeder Benutzer muss seine eigenen API Keys beantragen und ubergeben
- **OAuth Token gultig fur ca. ein Jahr:** Nach Ablauf des `CLAUDE_CODE_OAUTH_TOKEN` muss erneut `claude setup-token` ausgefuhrt werden, um einen neuen Token zu erhalten
- **Datenpersistenz:** Gesprachsverlauf und Erinnerungen im Container gehen beim Loschen des Containers verloren. Fur Persistenz mounten Sie ein Volume: `-v openclaw-data:/home/coder/.openclaw`
- **Sicherheit:** Bitte befolgen Sie alle Empfehlungen aus XIV. Sicherheitshinweise. Der Docker-Container bietet zwar eine Isolationsschicht, aber das Risiko eines API Key-Leaks erfordert weiterhin Aufmerksamkeit

XVI. Fazit und Ausblick

Von der Installation bis zum Abschluss der sechs Praxisszenarien hat der gesamte Prozess weniger als eine Stunde gedauert. OpenClaw hat tatsachlich den „autonomen KI-Agenten" von einem abstrakten Konzept in ein sofort einsatzbereites Werkzeug verwandelt — ein Befehl zur Installation, ein QR-Code-Scan zum Verbinden des Smartphones, und dann per naturlicher Sprache die KI steuern.

Wir mussen jedoch realistisch bleiben: **Die Spannung zwischen Benutzerfreundlichkeit und Sicherheit** ist derzeit der grosste Widerspruch von OpenClaw. Es lasst die KI Ihren gesamten Computer steuern — das ist zugleich sein grosster Vorteil und sein grosstes Risiko. Im aktuellen Stadium, in dem die Sicherheitsarchitektur noch nicht vollstandig ausgereift ist, empfehlen wir, OpenClaw als **Lern- und Experimentierwerkkzeug** zu betrachten und nicht als Produktivitats-Kernkomponente.

Das Zeitalter der KI-Agenten hat begonnen — daran besteht kein Zweifel. Die virale Verbreitung von OpenClaw beweist die enorme Nachfrage des Marktes nach „KI-Automatisierung". Unabhangig davon, ob Sie OpenClaw langfristig nutzen, wird das Durcharbeiten dieses Tutorials — und das personliche Erleben der Funktionsweise von KI-Agenten — Ihnen helfen, die Entwicklungsrichtung der KI-Branche in den nachsten ein bis zwei Jahren tiefgreifender zu verstehen.

Wenn Ihr Unternehmen die Einfuhrung von KI-Agenten und Automatisierungstools evaluiert, laden wir Sie herzlich ein, mit unserem Forschungsteam ein vertieftes Gesprach zu fuhren. Wir werden die Entwicklung von OpenClaw und ahnlichen Produkten weiterhin verfolgen und unseren Kunden helfen, in einer Ara der bluhenden Tool-Vielfalt die am besten geeigneten Technologieentscheidungen zu treffen.

References

1. Growth Foundry. (2026). *From 9,000 to 157,000 Stars in 60 Days: The OpenClaw Viral Growth Case Study.* Growth Foundry. growth.maestro.onl

2. CNBC. (2026). *From Clawdbot to Moltbot to OpenClaw: The rise and controversy of the open-source AI agent.* CNBC Technology. cnbc.com

3. Scientific American. (2026). *OpenClaw is an open-source AI agent that runs your computer.* Scientific American. scientificamerican.com

4. OpenClaw Documentation. (2026). *Getting Started — OpenClaw Official Docs.* docs.openclaw.ai

5. Pragmatic Engineer. (2026). *The creator of Clawd: "I ship code I don't read."* The Pragmatic Engineer Newsletter. pragmaticengineer.com

6. The Hacker News. (2026). *OpenClaw Bug Enables One-Click Remote Code Execution (CVE-2026-25253).* The Hacker News. thehackernews.com

7. CrowdStrike. (2026). *What Security Teams Need to Know About OpenClaw.* CrowdStrike Blog. crowdstrike.com

8. Cisco Blog. (2026). *Personal AI Agents like OpenClaw Are a Security Nightmare.* Cisco Blogs. blogs.cisco.com