# Anleitung zur Installation und Nutzung von Claude-Flow v2 (Hive-Mind Modus)

In dieser Anleitung zeigen wir Schritt für Schritt, wie du **Claude-Flow v2** installierst und in einem Projekt im **Hive-Mind Modus** nutzt. Wir gliedern das Vorgehen in vier Bereiche:

1. **Globales Setup:** Einrichten der Umgebung (WSL/Ubuntu, Node.js, Claude Code und Claude-Flow global installieren).
2. **Projekt-Setup:** Einrichtung in deinem Projektordner (Repo) inklusive Projektinitialisierung.
3. **Agenten-Konfiguration:** Auswahl und Vorkonfiguration der KI-Agenten (z.B. vorgefertigte Agentenrollen von GitHub nutzen).
4. **Start im Hive-Mind Modus:** Starten von Claude-Flow im Schwarmmodus (Hive-Mind) innerhalb des Projektordners mit den konfigurierten Agenten.

*(Hinweis: Wir gehen davon aus, dass du unter Windows mit WSL2 (Ubuntu) oder direkt unter Linux arbeitest, da dies für Claude-Flow empfohlen ist. Windows-Nutzer sollten WSL2 verwenden, da der direkte Windows-Support von Claude-Flow Alpha eingeschränkt ist*[*[1]*](https://github.com/aviadkim/pdf/blob/3bccdd731da10741911a09e9b456d6ecba9a096e/install-claude-flow.bat#L82-L87)*.)*

## 1. Allgemeines Setup (Global)

Zunächst richten wir die globale Umgebung ein. Dazu gehören Node.js, npm, Claude Code (Anthropic CLI) sowie Claude-Flow selbst. Führe diese Schritte *auf Systemebene* (z.B. in WSL2 Ubuntu oder Linux Terminal) aus:

* **Node.js 18+ und npm installieren:** Stelle sicher, dass Node.js (Version 18 oder höher) und npm verfügbar sind[[2]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,Windows%20Installation%20for%20special%20instructions). Überprüfe die Versionen im Terminal:
* node --version # sollte 18.x oder höher ausgeben  
  npm --version # sollte 9.x oder höher ausgeben
* Aktualisiere npm gegebenenfalls mit npm install -g npm@latest[[3]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=node%20). Bereinige evtl. den npm-Cache: npm cache clean --force[[4]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,g%20npm%40latest).
* **Claude Code global installieren:** Claude-Flow baut auf Anthropic **Claude Code** auf, daher muss zunächst das Claude Code CLI installiert und aktiviert werden[[5]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=%E2%9A%A0%EF%B8%8F%20IMPORTANT%3A%20Claude%20Code%20must,be%20installed%20first). Installiere es global mit npm und führe die Initialisierung aus:
* npm install -g @anthropic-ai/claude-code # Claude Code CLI installieren[6]  
  claude --version # Version prüfen (optional)[7]  
  claude --dangerously-skip-permissions # Initialisiere Claude Code (Berechtigungen bestätigen)[8]
* *Hinweis:* Der Parameter --dangerously-skip-permissions erlaubt Claude Code den Zugriff auf Dateien/Ordner ohne jede einzelne Anfrage zu bestätigen[[6]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,code). Nutze ihn mit Bedacht.
* **Claude-Flow (Alpha) global installieren:** Nachdem Claude Code läuft, installiere Claude-Flow v2 (Alpha-Version) global:
* npm install -g claude-flow@alpha # Claude-Flow Alpha global installieren[9]  
  claude-flow --version # Installation prüfen (sollte Version v2.0.0-alpha.xx zeigen)[10]
* Falls du Probleme unter Windows hast, ziehe in Betracht, stattdessen WSL2 zu nutzen (die Windows-Batch-Skripte versuchen bei Fehlern eine lokale Installation, was wir vermeiden möchten)[[11]](https://github.com/aviadkim/pdf/blob/3bccdd731da10741911a09e9b456d6ecba9a096e/install-claude-flow.bat#L80-L87). Durch WSL2/Linux umgehen wir diese Probleme.
* **Anthropic API Key setzen:** Claude-Flow benötigt einen gültigen Anthropic API-Schlüssel (z.B. von Claude.ai oder Anthropic Account). Exportiere den API-Key als Umgebungsvariable, damit Claude Code/Flow ihn nutzen kann[[12]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=1):
* export ANTHROPIC\_API\_KEY="dein-API-Schluessel"
* Füge diese Zeile auch deiner ~/.bashrc hinzu, damit der Key in neuen Terminals automatisch gesetzt wird[[13]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=export%20ANTHROPIC_API_KEY%3D%22your).
* **MCP-Integration (optional):** *MCP (Model Context Protocol)* ermöglicht Claude-Flow als Backend-Server in Claude Code zu registrieren. Dies ist optional, aber hilfreich, wenn du Claude Code interaktiv (z.B. in VSCode) verwendest. Du kannst Claude-Flow als MCP-Server hinzufügen:
* claude mcp add claude-flow "npx claude-flow@alpha mcp start"  
  claude mcp list # überprüfen, ob Claude-Flow als Server eingetragen ist[14]
* Damit kann Claude Code künftig Befehle an Claude-Flow weiterreichen. (Details siehe Claude-Flow Doku[[14]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=MCP%20Server%20Setup%20).)

Zusammengefasst hast du nun die globale Infrastruktur eingerichtet: Node/npm laufen, Claude Code ist installiert und initialisiert, Claude-Flow ist global verfügbar, und dein API-Key ist konfiguriert. Damit bist du bereit, ein konkretes Projekt mit Claude-Flow aufzusetzen.

## 2. Projektbezogenes Setup im Repository-Ordner

Im nächsten Schritt richtest du Claude-Flow innerhalb deines Projektordners ein. Das bedeutet, die notwendigen Konfigurationsdateien und Verzeichnisse für Claude-Flow werden in deinem Repository angelegt. Gehe wie folgt vor:

* **Projektordner auswählen/erstellen:** Wechsle in den Ordner deines Software-Projekts (Repository), in dem du Claude-Flow verwenden möchtest. Falls es noch keinen dedizierten Ordner gibt, lege einen neuen an:
* mkdir mein-projekt && cd mein-projekt
* **Claude-Flow im Projekt initialisieren:** Führe im Projektordner die Projekt-Initialisierung aus. Claude-Flow bietet dafür einen Befehl, der eine Basisstruktur anlegt. Du kannst optional ein Template angeben, z.B. "full-stack" für ein Full-Stack-Entwicklungsprojekt[[15]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,project). Beispiel:
* claude-flow project init --template full-stack # Projekt mit Full-Stack Template initialisieren[15]
* *Hinweis:* Falls project init nicht verfügbar ist, kannst du claude-flow init verwenden – beachte aber, dass dies ggf. ein generisches Setup macht. Das --template full-stack legt typische Agenten und Strukturen für Full-Stack-Projekte an.
* **Verzeichnisstruktur prüfen:** Nach erfolgreicher Initialisierung legt Claude-Flow im Projekt u.a. versteckte Verzeichnisse wie .hive-mind/, .swarm/ und memory/ an[[16]](https://github.com/ruvnet/claude-flow#:~:text=,created%20during%20tasks). Diese enthalten Konfigurationsdateien (z.B. config.json), SQLite-Datenbanken für Sessions und Memory sowie eventuelle Checkpoints:
* .hive-mind/ – Konfiguration & SQLite Session-Daten[16]   
  .swarm/ – Memory-Datenbank (memory.db) für Swarm-Sessions[16]   
  memory/ – Speicher für agentenspezifische Memories[17]   
  coordination/ – Temporäre Dateien für koordinierte Workflows[17]
* Diese Ordner könnten zunächst leer aussehen, da viel in SQLite-DBs gespeichert wird. Du kannst den Status prüfen mit Befehlen wie claude-flow memory stats oder claude-flow hive-mind status[[18]](https://github.com/ruvnet/claude-flow#:~:text=Don%27t%20panic%20if%20directories%20seem,show%20files%20in%20directory%20listings).
* **Projektname (optional):** Du kannst bei der Initialisierung einen Projektnamen vergeben, der in den Sessions genutzt wird. Beispiel:
* npx claude-flow@alpha init --force --project-name "mein-projekt"[19]
* Dies ist besonders bei größeren Projekten hilfreich, um mehrere Feature-„Hives” unterscheiden zu können.

Nach diesem Schritt ist dein Projekt-Repository Claude-Flow-fähig. Die notwendigen Dateien und Einstellungen sind jetzt lokal vorhanden.

*(Falls du mehrere getrennte Projekte hast, wiederhole diesen Schritt für jeden Projektordner separat. Für ein neues Projekt einfach neuen Ordner anlegen und erneut initialisieren*[*[20]*](https://github.com/ruvnet/claude-flow#:~:text=New%20feature%20in%20same%20project,flow%40alpha%20init)*.)*

## 3. Festlegung der Agenten (Agenten-Setup)

Claude-Flow folgt dem **Schwarm-Intelligenz-Ansatz**: Mehrere spezialisierte KI-Agenten arbeiten parallel zusammen, koordiniert von einer „Queen“-Agentin (Leitagentin). In Version 2.0 (Alpha) sind **64 vordefinierte spezialisierte Agenten** in **12 Kategorien** verfügbar[[21]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Claude,development%20tasks%20through%20coordinated%20collaboration). Diese decken diverse Rollen ab – von Entwicklung über Testing bis Architektur.

**Ziel dieses Schritts:** Festlegen, welche Agenten in deinem Projekt-Schwarm aktiv sein sollen. Du kannst dabei weitgehend auf die vorgefertigten Agenten zurückgreifen, die Claude-Flow mitliefert, anstatt alles selbst zu definieren.

* **Übersicht vordefinierter Agenten:** Die Agenten sind in Kategorien organisiert (mit YAML-Frontmatter + Markdown Doku in .claude/agents/...). Beispiele: Core Development (Coder, Reviewer, Tester, Planner, Researcher)[[22]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Type%20Description%20Priority%20,gathering%20and%20analysis%20specialist%20high), Swarm Coordination (verschiedene Koordinator-Typen wie hierarchisch, mesh)[[23]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Advanced%20coordination%20patterns%20for%20distributed,agent%20networks), Hive-Mind Intelligence (kollektive Intelligenz, Konsensbildung)[[24]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Type%20Description%20Capabilities%20%60collective,Distributed%20memory%20coordination%20memory_sync%2C%20context_sharing), GitHub Automation (PR-Manager, Issue-Tracker etc.)[[25]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Complete%20GitHub%20workflow%20automation%3A), SPARC Methodology (Specification, Pseudocode, etc.)[[26]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Type%20Description%20SPARC%20Phase,Iterative%20improvement%20and%20optimization%20Refinement) u.v.m.

Einen schnellen Eindruck der wichtigsten Rollen bietet auch die Projekt-README: Es gibt z.B. **Queen Agent** (Hauptkoordinator), **Architect Agents** (Architekturplanung), **Coder Agents** (Implementierung), **Tester Agents** (QA), **Analyst Agents** (Analyse), **Security Agents** (Security/Compliance), **DevOps Agents** (Deployment) usw.[[27]](https://github.com/ruvnet/claude-flow#:~:text=Intelligent%20Agent%20Types). Du musst also nicht jeden Agenten selbst schreiben – Claude-Flow bringt viele Spezialisten schon mit.

* **Agentenauswahl für dein Projekt:** Überlege, welche Rollen für dein Projekt relevant sind. Für viele Anwendungsfälle gibt es empfohlene „Schwarm-Zusammensetzungen“ (Swarm Patterns) in der Doku. Zum Beispiel:
* *Full-Stack Development Swarm (8 Agents):* Systemarchitekt, Backend-Entwickler, Mobile-Entwickler, DB-Designer (Coder), API-Dokumentator, CI/CD-Ingenieur, Performance-Tester, Produktionsvalidator[[28]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Task%28%22System%20architecture%22%2C%20%22...%22%2C%20%22system,validator).
* *Distributed System Swarm (6 Agents):* Byzantine-Coordinator, Raft-Manager, Gossip-Coordinator, CRDT-Synchronizer, Security-Manager, Performance-Monitor[[29]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=,agents).
* *GitHub Workflow Swarm (5 Agents):* PR-Manager, Code-Review-Swarm, Issue-Tracker, Release-Manager, Workflow-Automation[[30]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=,agents).
* *SPARC TDD Swarm (7 Agents):* Specification, Pseudocode, Architecture, SPARC-Coder (Implementierung), TDD London Tester, Refinement, Production-Validator[[31]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=,agents).

Diese Muster kannst du als Ausgangspunkt nehmen. Du könntest z.B. für ein **Python-Backend-Projekt** einen Schwarm bilden aus *System-Architect*, *Backend-Dev*, *Code-Reviewer*, *Tester*, *DevOps Engineer*, etc. Für ein **Android-App-Projekt** entsprechend *Mobile-Dev*, *UI/UX-Analyst*, *Backend-Dev* (falls ein Backend dabei ist), *Tester* etc. Die genaue Auswahl hängt von deinem Projekt ab – wähle 5–8 Kernagenten, die die wichtigsten Disziplinen abdecken (Claude-Flow empfiehlt für kritische Tasks 3–5 Agenten, für komplexere Features ~5–8 Agenten[[32]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=%2A%20High%20Priority%3A%20Use%203,8%2B%20agents%20with%20proper%20coordination)).

* **Agenten-Dateien anpassen (optional):** Die vordefinierten Agenten haben Standardbeschreibungen und Fähigkeiten (Capabilities). Du kannst sie bei Bedarf anpassen. Jede Agentendefinition befindet sich als Markdown mit YAML-Kopf in .claude/agents/<kategorie>/<agent-name>.md. Dort kannst du z.B. die description anpassen oder Hooks definieren. In der Regel reichen die Defaults aber aus. Wichtig ist vor allem, die richtigen **Agententypen auszuwählen**. Eine Übersicht aller Typen und Zwecke findest du in der Doku (z.B. developer für Coding, tester für Tests, architect für Architektur etc.)[[33]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Types).
* **Konfiguration speichern:** Idealerweise möchtest du die Auswahl deiner Agenten für dein Projekt dokumentieren oder speichern, damit du sie leicht beim Start laden kannst. Zwar lädt Claude-Flow nicht direkt eine „Agentenliste-Datei“, aber du kannst dir behelfen:
* *Möglichkeit A:* Lege eine Markdown oder JSON-Datei (z.B. claude\_settings/Agents\_Python.md) an, in der du notierst, welche Agentenrollen dein Schwarm umfasst (evtl. mit kurzer Beschreibung der Aufgabenverteilung). Diese dient als Referenz für dich oder Teammitglieder.
* *Möglichkeit B:* Verwende Claude-Flows **Workflow-Feature**, um komplexe Aufgaben inklusive Agentenzuweisungen vorzuplanen. Du könntest z.B. eine Workflow-Datei schreiben, in der einzelne Tasks bestimmten Agententypen zugewiesen sind (siehe Doku-Beispiel *Feature Development Workflow*, wo Tasks wie "requirements\_analysis" dem Agent analyst und "implementation" dem implementer zugewiesen werden[[34]](https://github.com/ruvnet/claude-flow/blob/02e1db0fa77a89114bafdc73ce5ca4f738d86983/docs/examples/README.md#L114-L123)[[35]](https://github.com/ruvnet/claude-flow/blob/02e1db0fa77a89114bafdc73ce5ca4f738d86983/docs/examples/README.md#L116-L125)). Solche Workflows lassen sich mit claude-flow workflow execute <file> laden[[36]](https://github.com/ruvnet/claude-flow/blob/02e1db0fa77a89114bafdc73ce5ca4f738d86983/docs/examples/README.md#L30-L33). Das ist zwar eher für Abfolgen von Tasks gedacht, kann aber helfen, einen vordefinierten Plan mit Agenten zu fahren.

**Besonderheit: Modellzuteilung (Opus vs Sonnet)** – Anthropic stellt unterschiedliche Modell-Backends bereit, z.B. **Claude Opus 4.1** (sehr leistungsfähig, hoher Kontext, teuer) und **Claude Sonnet 4** (schneller, günstiger, etwas limitierter)[[37]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=%2A%20Opus%204.1%2C%20%60claude). Ein guter Ansatz ist: **Queen-Agent auf Opus 4.1, Worker-Agenten auf Sonnet 4** zu setzen, um Kosten und Performance auszubalancieren. Claude-Flow unterstützt dynamisches Modell-Routing bereits mit einem speziellen *Model-Selector*-Agenten: *“Dieser Agent routet strategische Entscheidungen an Claude Opus für tiefes Reasoning, während Claude Sonnet für effiziente Ausführung genutzt wird.”*[[38]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L27-L35).

So kannst du das umsetzen:  
- **Standardmodell festlegen:** Setze die Umgebungsvariable ANTHROPIC\_MODEL auf das gewünschte **Standardmodell** für Claude Code. Beispielsweise könntest du Sonnet 4 als Default in der Shell definieren, damit alle normalen Agenten Sonnet nutzen[[39]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=For%20BASH%20users%20). Im ~/.bashrc:

export ANTHROPIC\_MODEL="claude-sonnet-4-20250514"

Dadurch nutzen alle Claude-Code-Aufrufe standardmäßig Sonnet 4 (sofern dein API-Zugang Sonnet erlaubt – *Anmerkung:* Opus-Modelle stehen nur Enterprise- oder bestimmten zahlenden Nutzern zur Verfügung[[40]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=%E2%9A%A0%EF%B8%8F%20Model%20Access%3A%20When%20using,able%20to%20use%20Opus%20models)). - **Queen-Agent mit Opus starten:** Für die leitende Agentin kannst du gezielt Opus verwenden. Starte z.B. Claude Code manuell mit dem Opus-Parameter oder nutze Claude-Flow’s Spawn-Option. In Claude Code selbst könntest du vor dem Hive-Start den Befehl /model Opus 4.1 eingeben, oder via CLI claude --model claude-opus-4-1-20250805 eine Session auf Opus öffnen[[41]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=2,to%20use%20for%20that%20session). Alternativ beim Start des Hive (siehe nächster Abschnitt) den **Queen-Spawn explizit mit Opus** anfordern. Claude-Flow bietet Flags wie --model=opus bei agent spawn oder --decision-model=opus bei swarm init[[42]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L125-L133)[[43]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L140-L148). Beispiel:

claude-flow agent spawn coordinator --model=opus --name "Queen" --task "Projektkoordination starten"

Danach könntest du weitere Agenten (ohne explizites Modell-Flag) spawnen, die dann das Default (Sonnet) nutzen.  
- **Dynamic Model Routing einschalten (optional):** Der elegantere Weg ist, das eingangs erwähnte Model-Selector-Feature zu aktivieren. Setze z.B.:

export CLAUDE\_FLOW\_DYNAMIC\_MODELS=true  
export CLAUDE\_FLOW\_OPUS\_ENDPOINTS="planning,analysis,architecture,debugging"  
export CLAUDE\_FLOW\_SONNET\_ENDPOINTS="implementation,testing,documentation,execution"

Damit versucht Claude-Flow automatisch, für strategische Aufgaben Opus zu nehmen und für Implementierungsaufgaben Sonnet[[44]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L48-L56). Du kannst diese Logik in der .claude/settings.json unter modelRouting konfigurieren (siehe Doku) oder den *Model-Selector Agent* in deinen Schwarm aufnehmen, der genau diese Aufgabe übernimmt.

Kurz gesagt: **Stelle sicher, dass die Queen (leitende KI) mit Claude Opus 4.1 läuft und alle anderen Agenten mit Claude Sonnet 4.0**, um eine gute Balance aus „Intelligenz“ und Effizienz zu haben. Die obigen Methoden bieten dafür verschiedene Ansätze – vom manuellen Setzen des Modells bis zum automatischen Routing.

## 4. Claude-Flow im Hive-Mind Modus starten (im Projektordner)

Nachdem alles installiert ist und die Agentenauswahl feststeht, folgt das eigentliche **Starten von Claude-Flow im Hive-Mind Modus** innerhalb deines Projekts. Der Hive-Mind Modus bedeutet, dass ein Schwarm von Agenten parallel an einer komplexen Aufgabe oder an mehreren Aufgaben koordiniert arbeitet – mit einer Queen an der Spitze. So gehst du vor:

* **Terminal vorbereiten:** Öffne ein Terminal (WSL oder Linux) und wechsle in den Projektordner, den du in Schritt 2 initialisiert hast. Stelle sicher, dass $ANTHROPIC\_API\_KEY und ggf. $ANTHROPIC\_MODEL richtig gesetzt sind (prüfe mit echo $ANTHROPIC\_API\_KEY).
* **Hive-Mind Wizard (optional):** Claude-Flow bietet einen interaktiven Wizard, der dir beim Start eines neuen Hives hilft. Du kannst ihn ausführen mit:
* npx claude-flow@alpha hive-mind wizard[45]
* Der Wizard fragt typischerweise nach dem Ziel/Projekt (Feature-Beschreibung), der gewünschten Topologie (hierarchical, mesh, etc.), Anzahl der Agenten oder speziellen Rollen und ähnlichem. Er erleichtert den Einstieg insbesondere, wenn du unsicher bist, welche Agenten du nehmen sollst.
* **Hive starten (spawn):** Hast du alle Einstellungen parat (mit oder ohne Wizard), starte den Schwarm durch Spawn eines Hive-Mind Prozesses. Im einfachsten Fall gibst du nur ein Ziel in natürlicher Sprache an; Claude-Flow wird dann entsprechende Agenten spawnen. Beispiel:
* npx claude-flow@alpha hive-mind spawn "Implementiere User-Authentifizierung im Projekt" --claude[46][47]
* Mit --claude stellst du sicher, dass Anthropic Claude genutzt wird (Standard) – das Flag ist vor allem da, falls mehrere Modellprovider unterstützt würden, hier aber unbedingt anzugeben laut Doku[[46]](https://github.com/ruvnet/claude-flow#:~:text=,claude). Claude-Flow erstellt nun eine neue Hive-Session für diese Aufgabe. Die Ausgabe im Terminal enthält meist eine **Session-ID** (z.B. session-20250812-abc123), die deinen Hive identifiziert.

*Tipp:* Du kannst beim Spawn bereits die Agenten festlegen. Nutze das Flag --agents. Entweder gibst du eine Zahl an (z.B. --agents 8 für 8 automatisch gewählte Agenten gemäß Standardstrategie) oder eine kommagetrennte Liste von Agententypen. Beispiel für einen custom Spawn mit bestimmten Agenten:

npx claude-flow@alpha hive-mind spawn "Start Android App Entwicklung" \  
 --agents system-architect,mobile-dev,backend-dev,tester,api-docs \  
 --topology hierarchical --claude

Hier würden 5 Agenten gespawnt: Architekt, Mobile-Entwickler, Backend-Entwickler, Tester und API-Dokumentator, in einer hierarchischen Topologie mit einem davon als Queen (Koordinator). Diese Möglichkeit erlaubt es, deine in Schritt 3 geplante Zusammensetzung direkt zu verwenden[[48]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=%23%20Use%20single%20agent%20claude,implement%20user%20authentication). (Ohne --agents wird beim swarm-Befehl eine Standard-Auswahl für den Task getroffen[[48]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=%23%20Use%20single%20agent%20claude,implement%20user%20authentication), im Hive-Mind Modus wirst du meist spezifischer vorgehen wollen.)

* **Hive-Mind Session nutzen:** Nach dem Start laufen die Agenten und koordinieren sich, um die Aufgabe(n) zu bearbeiten. Du kannst nun mit Claude-Flow interagieren, z.B.:
* Den Status abfragen: claude-flow hive-mind status zeigt aktive Agenten und ihren Fortschritt[[47]](https://github.com/ruvnet/claude-flow#:~:text=npx%20claude,claude)[[46]](https://github.com/ruvnet/claude-flow#:~:text=,claude).
* Ergebnisse einsehen oder weitere Befehle geben: Je nach Integration kannst du in Claude Code schauen, was die Agenten tun (sofern du es dort eingebunden hast), oder du lässt dir via CLI Logs/Ergebnisse ausgeben.
* **Weitere Tasks hinzufügen:** Du kannst dem laufenden Hive neue Anweisungen geben. Ein einfacher Weg ist der Befehl claude-flow swarm "<neue Teilaufgabe>" --continue-session, um im Kontext der aktuellen Session eine weitere Aufgabe parallel/sequentiell auszuführen[[49]](https://github.com/ruvnet/claude-flow#:~:text=,session)[[50]](https://github.com/ruvnet/claude-flow#:~:text=npx%20claude,session).
* **Sessions verwalten:** Hive-Mind Sessions sind persistent (dank SQLite Memory). Du kannst die Session pausieren und später fortsetzen. Nützliche Befehle:
* claude-flow hive-mind sessions listet alle Hives/Sessions auf.
* claude-flow hive-mind resume <session-id> setzt eine bestimmte Session fort[[51]](https://github.com/ruvnet/claude-flow#:~:text=,claude)[[52]](https://github.com/ruvnet/claude-flow#:~:text=,xxxxx).
* claude-flow memory query --recent zeigt kürzlich gelernte Fakten/Infos an, um zu sehen, was die Agenten im Memory haben[[53]](https://github.com/ruvnet/claude-flow#:~:text=npx%20claude,claude).
* claude-flow hive-mind status erinnert dich daran, woran gearbeitet wurde (Aufgabenhistorie)[[54]](https://github.com/ruvnet/claude-flow#:~:text=Continuing%20Previous%20Work).
* **Automatisierung via Skript (optional):** Da du eingangs "start\_claude\_flow\_v2.sh" erwähnt hast: Du kannst dir tatsächlich ein Start-Skript schreiben, das all das auf einmal erledigt. Dieses Skript könnte z.B. deinen API-Key setzen, ggf. das Modell definieren (Opus/Sonnet), dann in den Projektordner wechseln und den obigen Spawn-Befehl mit deinen gewünschten Agents ausführen. So startest du mit einem Klick immer deinen vordefinierten Schwarm. Bei Bedarf legst du mehrere Skripte an (z.B. start\_swarm\_python.sh, start\_swarm\_android.sh), die jeweils andere Agentensets laden. In diesen Skripten kannst du im Prinzip die hive-mind spawn ... --agents ... Befehle hinterlegen, wie oben gezeigt. Damit erzielst du die gewünschte **projektbezogene Schwarm-Zusammensetzung auf Knopfdruck**.

Nachdem der Hive-Mind gestartet ist, übernimmt Claude-Flow die Koordination. Die *Queen*-Agentin plant und verteilt Aufgaben an die Worker-Agenten. Durch die Parallelisierung und den ständigen Austausch erreichen die Agenten gemeinsam schneller Ergebnisse, als ein einzelner KI-Partner es könnte. Dieses **Schwarmsystem** ist die Stärke von Claude-Flow: *„Claude-Flow v2.0.0 führt eine revolutionäre Hive-Mind Architektur ein, in der eine Queen-AI spezialisierte Worker Agents in perfekter Harmonie koordiniert.”*[[55]](https://github.com/ruvnet/claude-flow#:~:text=Queen).

Du kannst nun zuschauen, wie Code geschrieben, Tests erzeugt, Dokumentation erstellt oder Analysen durchgeführt werden – alles im Teamwork der Agenten. Viel Erfolg mit deinem Claude-Flow Hive! 😊

### Quellen

* Claude-Flow GitHub – **Installation Guide** (Prerequisites, Installation)[[5]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=%E2%9A%A0%EF%B8%8F%20IMPORTANT%3A%20Claude%20Code%20must,be%20installed%20first)[[9]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=%23%203.%20Install%20Claude,flow%40alpha)
* Claude-Flow GitHub – **Installation Guide** (Project Initialization, Verification)[[15]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,project)[[16]](https://github.com/ruvnet/claude-flow#:~:text=,created%20during%20tasks)
* Claude-Flow GitHub – **README/Usage** (Swarm vs Hive-Mind, Workflow Patterns)[[45]](https://github.com/ruvnet/claude-flow#:~:text=,build%20me%20a%20REST%20API)[[56]](https://github.com/ruvnet/claude-flow#:~:text=,claude)
* Claude-Flow GitHub – **Agent System Overview** (Agent Categories, Swarm Patterns)[[22]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Type%20Description%20Priority%20,gathering%20and%20analysis%20specialist%20high)[[28]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Task%28%22System%20architecture%22%2C%20%22...%22%2C%20%22system,validator)
* Claude-Flow GitHub – **Model Selector (Opus/Sonnet)**[[38]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L27-L35)[[44]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L48-L56)
* Anthropic Help Center – **Claude Code Model Configuration** (Opus 4.1 vs Sonnet 4)[[37]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=%2A%20Opus%204.1%2C%20%60claude)[[39]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=For%20BASH%20users%20)

[[1]](https://github.com/aviadkim/pdf/blob/3bccdd731da10741911a09e9b456d6ecba9a096e/install-claude-flow.bat#L82-L87) [[11]](https://github.com/aviadkim/pdf/blob/3bccdd731da10741911a09e9b456d6ecba9a096e/install-claude-flow.bat#L80-L87) install-claude-flow.bat

<https://github.com/aviadkim/pdf/blob/3bccdd731da10741911a09e9b456d6ecba9a096e/install-claude-flow.bat>

[[2]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,Windows%20Installation%20for%20special%20instructions) [[3]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=node%20) [[4]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,g%20npm%40latest) [[5]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=%E2%9A%A0%EF%B8%8F%20IMPORTANT%3A%20Claude%20Code%20must,be%20installed%20first) [[6]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,code) [[7]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,code) [[8]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,version) [[9]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=%23%203.%20Install%20Claude,flow%40alpha) [[10]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=%23%20Install%20Claude,flow%40alpha) [[12]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=1) [[13]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=export%20ANTHROPIC_API_KEY%3D%22your) [[14]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=MCP%20Server%20Setup%20) [[15]](https://github.com/ruvnet/claude-flow/wiki/Installation-Guide#:~:text=,project) Installation Guide · ruvnet/claude-flow Wiki · GitHub

<https://github.com/ruvnet/claude-flow/wiki/Installation-Guide>

[[16]](https://github.com/ruvnet/claude-flow#:~:text=,created%20during%20tasks) [[17]](https://github.com/ruvnet/claude-flow#:~:text=,created%20during%20tasks) [[18]](https://github.com/ruvnet/claude-flow#:~:text=Don%27t%20panic%20if%20directories%20seem,show%20files%20in%20directory%20listings) [[19]](https://github.com/ruvnet/claude-flow#:~:text=%23%20Project,app) [[20]](https://github.com/ruvnet/claude-flow#:~:text=New%20feature%20in%20same%20project,flow%40alpha%20init) [[27]](https://github.com/ruvnet/claude-flow#:~:text=Intelligent%20Agent%20Types) [[45]](https://github.com/ruvnet/claude-flow#:~:text=,build%20me%20a%20REST%20API) [[46]](https://github.com/ruvnet/claude-flow#:~:text=,claude) [[47]](https://github.com/ruvnet/claude-flow#:~:text=npx%20claude,claude) [[49]](https://github.com/ruvnet/claude-flow#:~:text=,session) [[50]](https://github.com/ruvnet/claude-flow#:~:text=npx%20claude,session) [[51]](https://github.com/ruvnet/claude-flow#:~:text=,claude) [[52]](https://github.com/ruvnet/claude-flow#:~:text=,xxxxx) [[53]](https://github.com/ruvnet/claude-flow#:~:text=npx%20claude,claude) [[54]](https://github.com/ruvnet/claude-flow#:~:text=Continuing%20Previous%20Work) [[55]](https://github.com/ruvnet/claude-flow#:~:text=Queen) [[56]](https://github.com/ruvnet/claude-flow#:~:text=,claude) GitHub - ruvnet/claude-flow: Claude-Flow v2.0.0 Alpha represents a leap in AI-powered development orchestration. Built from the ground up with enterprise-grade architecture, advanced swarm intelligence, and seamless Claude Code integration.

<https://github.com/ruvnet/claude-flow>

[[21]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Claude,development%20tasks%20through%20coordinated%20collaboration) [[22]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Type%20Description%20Priority%20,gathering%20and%20analysis%20specialist%20high) [[23]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Advanced%20coordination%20patterns%20for%20distributed,agent%20networks) [[24]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Type%20Description%20Capabilities%20%60collective,Distributed%20memory%20coordination%20memory_sync%2C%20context_sharing) [[25]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Complete%20GitHub%20workflow%20automation%3A) [[26]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Type%20Description%20SPARC%20Phase,Iterative%20improvement%20and%20optimization%20Refinement) [[28]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Task%28%22System%20architecture%22%2C%20%22...%22%2C%20%22system,validator) [[29]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=,agents) [[30]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=,agents) [[31]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=,agents) [[32]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=%2A%20High%20Priority%3A%20Use%203,8%2B%20agents%20with%20proper%20coordination) [[33]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=Agent%20Types) [[48]](https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview#:~:text=%23%20Use%20single%20agent%20claude,implement%20user%20authentication) Agent System Overview · ruvnet/claude-flow Wiki · GitHub

<https://github.com/ruvnet/claude-flow/wiki/Agent-System-Overview>

[[34]](https://github.com/ruvnet/claude-flow/blob/02e1db0fa77a89114bafdc73ce5ca4f738d86983/docs/examples/README.md#L114-L123) [[35]](https://github.com/ruvnet/claude-flow/blob/02e1db0fa77a89114bafdc73ce5ca4f738d86983/docs/examples/README.md#L116-L125) [[36]](https://github.com/ruvnet/claude-flow/blob/02e1db0fa77a89114bafdc73ce5ca4f738d86983/docs/examples/README.md#L30-L33) README.md

<https://github.com/ruvnet/claude-flow/blob/02e1db0fa77a89114bafdc73ce5ca4f738d86983/docs/examples/README.md>

[[37]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=%2A%20Opus%204.1%2C%20%60claude) [[39]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=For%20BASH%20users%20) [[40]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=%E2%9A%A0%EF%B8%8F%20Model%20Access%3A%20When%20using,able%20to%20use%20Opus%20models) [[41]](https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration#:~:text=2,to%20use%20for%20that%20session) Claude Code Model Configuration | Anthropic Help Center

<https://support.anthropic.com/en/articles/11940350-claude-code-model-configuration>

[[38]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L27-L35) [[42]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L125-L133) [[43]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L140-L148) [[44]](https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md#L48-L56) model-selector.md

<https://github.com/bilalmachraa82/MariaIntelligence-1/blob/99c54c76cde66427f1ae49fbfc211fe49672c62e/.claude/agents/specialized/model-selector.md>