# Projekt-Setup und Dateistruktur

Um einen **Hive-Mind-Agentenschwarm** mit Claude Flow zu starten, legen Sie im Projektverzeichnis zunächst eine .claude-flow-Unterstruktur an. Darin gehört ein Ordner saved-configs, in dem die JSON-Konfigurationsdateien für Ihre Schwärme liegen. Beispielhafte Struktur:

| Pfad | Beschreibung |
| --- | --- |
| <ProjektRoot>/ | Arbeitsverzeichnis Ihres Projekts |
| <ProjektRoot>/.claude-flow/ | Claude-Flow-Projektkonfiguration (v2) |
| <ProjektRoot>/.claude-flow/config.json | (Optional) Spiegel der aktuellen Schwarm-Konfiguration (siehe unten) |
| <ProjektRoot>/.claude-flow/saved-configs/ | Ordner mit gespeicherten Schwarm-Konfigurationen |
| <ProjektRoot>/.claude/ | (v1-v2 Migration) Älterer Konfigurationspfad für Claude-Code |
| <ProjektRoot>/.claude/config.json | (Optional) Version der Konfiguration im .claude-Ordner (siehe unten) |
| **Beispiel**: .claude-flow/saved-configs/MySwarm.json | JSON-Konfigurationsdatei für Schwarm “MySwarm” |

In der .claude-flow/saved-configs/-Datei speichern Sie ein JSON-Objekt, das den Schwarm definiert. Die App kann beim Speichern der Einstellungen (z.B. über save\_complete\_config()) diese Datei erzeugen. Zur Kompatibilität schreiben manche GUIs oder Skripte dieselbe Konfiguration zusätzlich in .claude/config.json (Projektordner) und/oder .claude-flow/config.json. Claude Flow selbst unterstützt standardmäßig den --config-Pfadparameter[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable), richtet sich aber oft auch nach einer Datei claude-flow.config.json oder .claude/config.json im Projekt. In Version 1.x wurde etwa eine claude-flow.config.json bzw. .claude/config.json erzeugt[[2]](https://www.npmjs.com/package/claude-flow/v/1.0.70#:~:text=,building%20focus). Wir empfehlen, bei Bedarf die JSON-Datei aus saved-configs nach .claude/config.json zu kopieren, damit sie von allen Komponenten gefunden wird.

# Beispiel: JSON-Schwarm-Konfiguration

Die JSON-Datei beschreibt die ausgewählten Agenten, Topologie und Aufgabe. Ein vollständiges Beispiel könnte so aussehen:

{  
 "project": {  
 "name": "MyProject",  
 "namespace": "alpha"  
 },  
 "agents": {  
 "selected": ["queen", "backend-dev", "frontend-dev", "system-architect", "tester"],  
 "queen\_model": "anthropic/claude-2",  
 "worker\_model": "anthropic/claude-2"  
 },  
 "swarm": {  
 "topology": "mesh",  
 "task": "Bearbeite alle offenen Issues im Git-Repository"  
 },  
 "settings": {  
 "memorySize": "1GB",  
 "parallelExecution": true  
 },  
 "codex\_mode": false  
}

* **project.\*** – Projektmetadaten (Name, Namespace o.ä., optional).
* **agents.selected** – Liste der Agententypen im Schwarm (z.B. "queen", "coder", "tester").
* **agents.queen\_model**, **agents.worker\_model** – Modellversion für übergeordnete bzw. normale Agenten.
* **swarm.topology** – Schwarm-Topologie (z.B. "mesh", "hierarchical", "ring", "star").
* **swarm.task** – Die Aufgabenbeschreibung (string) für den Schwarm.
* **settings.\*** – Weitere Schwarm- oder Leistungseinstellungen (z.B. Größe des Speichers, Timeout).
* **Optionale Felder** wie "codex\_mode" oder "agentCapabilities" können von spezifischen Implementierungen genutzt werden (die GUI speichert z.B. codex\_\*-Einstellungen).

Ein vollständiger Schwarm wird von der CLI oder der App anhand dieser Datei initialisiert. (Ein offizielles JSON-Schema ist nicht veröffentlicht, aber die oben genannten Schlüssel orientieren sich am internen Format.)

# Integration in die Claude-Flow CLI

Nachdem die Konfigurationsdatei geschrieben ist, starten Sie Claude Flow im Hive-Mind-Modus mit dem Befehl:

npx claude-flow@alpha hive-mind spawn "<Aufgabe>" --config ".claude-flow/saved-configs/MySwarm.json" --verbose

* **hive-mind spawn "<Aufgabe>"**: Startet einen neuen Hive-Mind-Schwarm mit der übergebenen Aufgabenbeschreibung[[3]](https://github.com/ruvnet/claude-flow#:~:text=,claude).
* **--config <Pfad>**: Gibt den Pfad zur JSON-Konfigurationsdatei an[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable). Wenn der Parameter weggelassen wird, sucht Claude Flow ggf. nach einer Standarddatei (.claude/config.json oder claude-flow.config.json).
* **--verbose**: Aktiviert ausführliche Protokollausgabe im Terminal (sichtbar für Debugging)[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable).
* **Weitere Optionen**: Sie können z.B. --agents 3 überschreiben oder --namespace <ns> angeben. Beispiele aus der Dokumentation:
* claude-flow hive-mind spawn "Implement user authentication" --claude[[3]](https://github.com/ruvnet/claude-flow#:~:text=,claude)
* claude-flow hive-mind spawn "auth-system" --namespace auth --claude[[4]](https://github.com/ruvnet/claude-flow#:~:text=,claude)
* claude-flow hive-mind spawn "Research microservices patterns" --agents researcher,analyst --claude[[5]](https://github.com/ruvnet/claude-flow#:~:text=,claude)

In Ihrem App-Code rufen Sie diesen Befehl via spawn-Prozess aus. Achten Sie darauf, das Projektverzeichnis als Arbeitsverzeichnis zu verwenden, damit die relativen Pfade passen. Der Parameter --verbose ist hilfreich, um alle Aktivitäten im Terminal mitzuverfolgen (Statusmeldung, Logs usw.).

# Konfigurationsdatei automatisch bereitstellen

Falls die CLI-Invocation den --config-Pfad nicht unterstützt oder Sie auf Nummer sicher gehen wollen, kopieren Sie die JSON-Konfiguration vor dem Start nach .claude/config.json im Projektstamm. In Version 1 wurde diese Datei bei init automatisch erstellt[[2]](https://www.npmjs.com/package/claude-flow/v/1.0.70#:~:text=,building%20focus). Beispiel:

cp .claude-flow/saved-configs/MySwarm.json .claude/config.json  
claude-flow hive-mind spawn "<Aufgabe>" --verbose

Ihr App-Code kann prüfen, ob --config verfügbar ist (z.B. über --help) und andernfalls die Datei duplizieren. Die config-Option der CLI ist jedoch eine saubere Lösung: Sie ist offiziell dokumentiert und umgeht Mehrdeutigkeiten[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable).

# Agentenfähigkeiten

Die **Fähigkeiten** (Capabilities) der Agenten sind Teil der Agententyp-Definitionen und müssen nicht extra in der Schwarmkonfiguration gespeichert werden. In Claude Flow werden Agententypen üblicherweise über YAML/Markdown in .claude/agents/ definiert (mit Namen, Beschreibungen und Fähigkeiten). Wenn Sie Agenten über die JSON-Konfigurationsdatei spezifizieren, wird implizit auf diese Definitionen zugegriffen.

Lediglich spezialisierte Befehle oder Tools, die Agents ausführen können (z.B. Shell-Kommandos, GPT-Aufrufe), müssen korrekt in den Agentendefinitionen hinterlegt sein. Sie sind aber **nicht Teil** des oben gezeigten JSON-Schwarm-Objekts. Die Konfigurationsdatei steuert nur Auswahl, Topologie und globale Einstellungen, nicht die internen Plugin-Fähigkeiten der Agenten.

# Standardisierte Aufgaben und Vorlagen (GUI)

Für häufige Aufgaben oder Templates empfiehlt es sich, eine eigene JSON-Datei mit Task-Vorlagen zu pflegen. Beispielstruktur für eine tasks.json im Projekt:

[  
 {  
 "name": "Open Issues bearbeiten",  
 "description": "Bearbeite alle offenen GitHub Issues und dokumentiere die Ergebnisse",  
 "task": "Bearbeite alle offenen Issues im Repository",  
 "agents": ["coder", "tester", "reviewer"]  
 },  
 {  
 "name": "Neues Feature X",  
 "description": "Implementiere neues Feature X inklusive Tests",  
 "task": "Implementiere neues Feature X mit vollständiger Testabdeckung",  
 "agents": ["system-architect", "coder", "tester"]  
 }  
]

Die GUI kann solche Vorlagen laden (z. B. über ein Dropdown) und die Felder ins JSON-Format übernehmen. Sie speichert Änderungen an Aufgaben dann analog zu den Schwarm-Konfigurationen. In Ihrer App können Sie das JSON-Template-Verzeichnis ebenfalls durchsuchen und in die GUI-Liste einbinden. Das Format ist frei wählbar (JSON, YAML, Datenbank), wichtig ist nur, dass Name, Beschreibung und Task-Text sauber trennbar sind. Eine Tabelle z.B. mit den folgenden Spalten könnte helfen:

| Name | Beschreibung | Task-Text |
| --- | --- | --- |
| Open Issues | GitHub-Issues abarbeiten | Bearbeite alle offenen GitHub Issues. |
| Neues Feature X | Feature X mit Tests erstellen | Implementiere Feature X mit Tests. |

Beim Auswählen einer Vorlage füllt die App das Eingabeformular, ggf. inkl. Agentenvorschlag. Anschließend speichert sie den Schwarm wie oben beschrieben.

# Zusammenfassung und Befehlsübersicht

* **Dateien:** Legen Sie Ihr Schwarm-JSON in .claude-flow/saved-configs/\*.json ab. Optional kopieren Sie es nach .claude/config.json (für ältere CLI-Setups)[[2]](https://www.npmjs.com/package/claude-flow/v/1.0.70#:~:text=,building%20focus).
* **JSON-Format:** Enthält Schlüssel wie agents.selected, agents.queen\_model, swarm.topology, swarm.task sowie settings.\*. Beispiel siehe oben.
* **CLI-Start:** Verwenden Sie claude-flow hive-mind spawn "<Task>" --config <Datei> --verbose[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable)[[3]](https://github.com/ruvnet/claude-flow#:~:text=,claude). Sie können zusätzlich Agentenzahl (--agents), Namespace (--namespace) usw. setzen.
* **Terminal-Ausgabe:** Mit --verbose sehen Sie sämtliche Log-Ausgaben live im Terminal[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable).
* **Vorlagenspeicher:** Definieren Sie Aufgaben-Templates in einer separaten JSON-Datei und laden Sie diese in Ihrer GUI.

Indem Ihre App diese Schritte automatisiert, startet Claude Flow den Hive-Mind mit Ihrer gewünschten Konfiguration direkt aus dem Projekt heraus. Alle relevanten Aktivitäten und Ausgaben erscheinen dabei im Terminal (da --verbose gesetzt ist)[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable)[[3]](https://github.com/ruvnet/claude-flow#:~:text=,claude). So erhält der Benutzer einen durchgängigen Workflow vom Auswählen einer Aufgabe bis zum automatischen Start des Agentenschwarms.

**Quellen:** Claude Flow CLI-Dokumentation (Optionen und Beispiele)[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md#:~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable)[[3]](https://github.com/ruvnet/claude-flow#:~:text=,claude). (Zusätzlich wurden internen App-Dokumentationen entnommen, u.a. zu den Schlüsseln von saved-configs.)

[[1]](https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md" \l ":~:text=Flow%20operations.%20%60%60%60bash%20claude,Disable) raw.githubusercontent.com

<https://raw.githubusercontent.com/wiki/ruvnet/claude-flow/API-Reference.md>

[[2]](https://www.npmjs.com/package/claude-flow/v/1.0.70#:~:text=,building%20focus) claude-flow - npm

<https://www.npmjs.com/package/claude-flow/v/1.0.70>

[[3]](https://github.com/ruvnet/claude-flow#:~:text=,claude) [[4]](https://github.com/ruvnet/claude-flow#:~:text=,claude) [[5]](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>