

# Technische Details zu claude-flow@alpha Version 86

Claude-flow@alpha (V86) ist eine Enterprise-Grade AI-Orchestrierungsplattform mit Version 2.0.0-alpha.86, die revolutionäre Swarm-Intelligence-Koordination für Multi-Agent-Entwicklung bietet.

[GitHub +5](#) Die Konfigurationen werden primär in **JSON-Format** in mehreren hierarchisch organisierten Dateien gespeichert, wobei `.claude/settings.json` als Hauptkonfigurationsdatei dient, ergänzt durch

`.mcp.json` für MCP-Server und `claude-flow.config.json` für plattformspezifische Einstellungen. [GitHub](#)

[GitHub](#) Das System umfasst **64 spezialisierte AI-Agenten** verteilt auf 16 Kategorien mit **87 MCP-Tools**

[GitHub](#) [npm](#) und nutzt eine SQLite-Datenbank für persistente Cross-Session-Memory mit 12 spezialisierten Tabellen. [GitHub +2](#)

## Speicherorte der Agentenkonfigurationen

Die Agentenkonfigurationen und Presets für claude-flow@alpha (V86) folgen einer klaren Verzeichnisstruktur. Das System erstellt bei der Initialisierung automatisch die erforderlichen Verzeichnisse und Konfigurationsdateien. [GitHub +2](#)

```

project-root/
├── .claude/
│   ├── settings.json      # Hauptkonfigurationsdatei
│   ├── settings.local.json # Lokale Überschreibungen (gitignored)
│   └── commands/          # Dokumentation für alle 87 Tools
│       ├── analysis/      # Analyse-Command-Dokumentation
│       ├── automation/    # Automatisierungs-Commands
│       ├── coordination/   # Koordinations-Commands
│       ├── github/        # GitHub-Integration-Commands
│       ├── hooks/         # Hook-Command-Dokumentation
│       ├── memory/        # Memory-Management-Commands
│       └── workflows/     # Workflow-Commands
├── .mcp.json              # MCP-Server-Konfiguration (Projektebene)
├── claude-flow.config.json # Claude-flow spezifische Einstellungen
├── CLAUDE.md              # Projektkontext für AI
├── .hive-mind/            # Hive-Session-Daten
│   ├── config.json        # Hive-Mind-Konfiguration
│   └── sessions/          # Aktive Session-Daten
├── .swarm/                # Swarm-Koordinationsdaten
│   └── memory.db          # SQLite Memory-Datenbank
├── agents/                # 64+ spezialisierte Agenten-Configs
│   ├── architect.json
│   ├── coder.json
│   ├── researcher.json
│   ├── tester.json
│   └── [weitere 60+ Agenten-Konfigurationen]
├── swarms/                # Swarm-spezifische Konfigurationen
│   ├── development/
│   ├── testing/
│   └── production/
├── workflows/             # Automatisierungs-Workflows
│   ├── ci-cd/
│   └── deployment/

```

Die **27+ neuronalen Modelle** werden mit WASM SIMD-Beschleunigung in separaten Modulen verwaltet.

(npm) Die persistente Memory-Datenbank `.swarm/memory.db` speichert Cross-Session-Daten mit **12 spezialisierten Tabellen** für verschiedene Datentypen und Koordinationsmuster. (GitHub +3)

## Konfigurationsformat und Struktur

Das System verwendet primär **JSON** als Konfigurationsformat mit optionaler YAML-Unterstützung für bestimmte Workflow-Definitionen. Die Konfigurationsdateien folgen einer hierarchischen Struktur mit klaren Namespaces für verschiedene Funktionsbereiche.

### Hauptkonfigurationsdatei (.claude/settings.json)

```
json

{
  "model": "sonnet",
  "permissions": {
    "allow": [
      "Bash(mkdir:*)",
      "Bash(npm:*)",
      "Write",
      "Edit",
      "MultiEdit"
    ],
    "deny": []
  },
  "hooks": {
    "PreToolUse": [
      {
        "matcher": "Bash",
        "hooks": [
          {
            "type": "command",
            "command": "npx claude-flow@alpha hooks pre-command --command \"{\\}\" --validate-safety true"
          }
        ]
      }
    ],
    "PostToolUse": [
      {
        "matcher": "Write|Edit|MultiEdit",
        "hooks": [
          {
            "type": "command",
            "command": "npx claude-flow@alpha hooks post-edit --file \"{\\}\" --memory-key \"{swarm/{agent}/{step}\\}\""
          }
        ]
      }
    ]
  },
  "env": {
    "BASH_DEFAULT_TIMEOUT_MS": "300000",
    "BASH_MAX_TIMEOUT_MS": "600000"
  }
}
```

## MCP-Server-Konfiguration (.mcp.json)

json

```
{
  "mcpServers": {
    "claude-flow": {
      "command": "npx",
      "args": ["-y", "claude-flow@alpha", "mcp", "start"],
      "env": {}
    },
    "ruv-swarm": {
      "command": "npx",
      "args": ["-y", "ruv-swarm@latest", "mcp", "start"],
      "env": {}
    }
  }
}
```

Die Konfigurationsdateien unterstützen **Schema-Validierung** mit eingebauten Prüfungen für Parametertypen, Wertebereiche und erforderliche Felder. Das System führt automatische Kompatibilitätsprüfungen beim Start durch.

## Konkrete Beispielkonfiguration für Produktion

Eine vollständig ausgefüllte Produktionskonfiguration demonstriert die umfassenden Möglichkeiten des Systems mit realen Werten.

### Production-Ready claude-flow.config.json

json

```
{
  "name": "enterprise-development-system",
  "version": "2.0.0-alpha.86",
  "orchestrator": {
    "maxAgents": 12,
    "maxConcurrentAgents": 8,
    "defaultTopology": "hierarchical",
    "strategy": "development",
    "memoryEnabled": true,
    "faultTolerance": {
      "strategy": "retry-with-learning",
      "maxRetries": 3,
      "byzantineFaultTolerance": true,
      "healthCheckInterval": 30000
    }
  },
  "agents": {
    "types": [
      "queen", "architect", "coder", "reviewer",
      "tester", "security", "devops", "analyst",
      "researcher", "coordinator", "performance-benchmarker"
    ],
    "spawning": {
      "autoSpawn": true,
      "maxAge": "2h",
      "healthCheck": true,
      "batchSize": 5
    },
    "specialization": {
      "coder": {
        "verification": ["compile", "test", "lint", "typecheck"],
        "truthThreshold": 0.95,
        "languages": ["typescript", "python", "rust"],
        "maxFilesPerOperation": 10
      },
      "reviewer": {
        "verification": ["code-analysis", "security-scan", "performance-check"],
        "truthThreshold": 0.95,
        "reviewDepth": "comprehensive"
      },
      "tester": {
        "verification": ["unit-tests", "integration-tests", "coverage-check"],
        "truthThreshold": 0.85,
        "coverageTarget": 95
      }
    }
  }
}
```

```
},
"memory": {
  "backend": "sqlite",
  "persistentSessions": true,
  "database": ".swarm/memory.db",
  "tables": 12,
  "cacheSizeMB": 200,
  "compression": true,
  "distributedSync": true,
  "namespaces": ["default", "sparc", "neural", "coordination"],
  "retentionDays": 30
},
"neural": {
  "enabled": true,
  "models": 27,
  "wasmSimd": true,
  "training": {
    "patterns": ["coordination", "cognitive-analysis", "task-optimization"],
    "epochs": 50,
    "learningRate": 0.001,
    "batchSize": 32
  }
},
"hooks": {
  "enabled": true,
  "types": [
    "pre-task", "post-task", "pre-edit", "post-edit",
    "pre-command", "post-command", "session-start",
    "session-end", "pre-search", "post-search",
    "pre-analysis", "post-analysis", "error-recovery", "notify"
  ],
  "automation": {
    "agentAssignment": true,
    "performanceTracking": true,
    "errorRecovery": true,
    "autoFormat": true,
    "testOnSave": true
  }
},
"performance": {
  "parallelExecution": true,
  "tokenOptimization": true,
  "batchProcessing": true,
  "timeout": 300000,
  "maxOutputSize": 500000,
  "tokenLimit": 100000
},
```

```
"security": {
  "monitoring": true,
  "cryptographicSigning": true,
  "auditTrail": true,
  "sandboxing": true
},
"telemetry": {
  "enabled": true,
  "tokenTracking": true,
  "costAnalysis": true,
  "realTimeMonitoring": true,
  "exportFormat": "json"
}
}
```

Die Konfiguration zeigt **84.8% Erfolgsrate** bei SWE-Bench-Tests mit **2.8-4.4x schnellerer Ausführung** durch parallele Koordination und **32.3% Token-Reduktion** durch intelligente Optimierung. [GitHub +2](#)

## Definierte Parameter im Detail

Das Parametersystem von `claude-flow@alpha (V86)` umfasst mehrere Hauptkategorien mit spezifischen Einstellungsmöglichkeiten für verschiedene Anwendungsfälle.

### Orchestrator-Parameter

Die Orchestrator-Sektion steuert die grundlegende Koordinationslogik:

- **maxAgents** (1-15): Maximale Anzahl gleichzeitiger Agenten
- **maxConcurrentAgents** (1-12): Parallele Ausführungslimits
- **topology**: Netzwerkstruktur ("hierarchical", "mesh", "ring", "star", "sequential")
- **strategy**: Aufgabenverteilung ("balanced", "development", "parallel")
- **memoryEnabled**: Aktiviert persistente Memory-Funktionen
- **faultTolerance**: Byzantine Fault Tolerance für Enterprise-Zuverlässigkeit

### Agent-Parameter

Die Agent-Konfiguration definiert spezialisierte Rollen und Verhaltensweisen:

- **types**: Array mit 64 verfügbaren Agententypen aus 16 Kategorien
- **spawning.autoSpawn**: Automatische Agentenerstellung bei Bedarf
- **spawning.maxAge**: Lebensdauer der Agenten (Duration-String)
- **spawning.healthCheck**: Kontinuierliche Gesundheitsüberwachung
- **specialization**: Rollenspezifische Konfigurationen mit Verifikationsschwellenwerten

## Memory-System-Parameter

Das persistente Memory-System nutzt SQLite mit erweiterten Features:

- **backend**: Datenbank-Engine (standardmäßig "sqlite")
- **persistentSessions**: Cross-Session-Datenerhaltung
- **database**: Pfad zur SQLite-Datenbank
- **tables**: Anzahl spezialisierter Tabellen (standardmäßig 12)
- **cacheSizeMB**: Memory-Cache-Größe in Megabyte
- **compression**: Speicheroptimierung durch Kompression
- **distributedSync**: Multi-Instanz-Synchronisation
- **namespaces**: Organisierte Memory-Zugriffsbereiche

## Hook-System-Parameter

Das Hook-System bietet **14 Lifecycle-Management-Hooks**: [GitHub](#)

- **Pre-Task-Hooks**: Vorbereitung und Planung
- **Post-Task-Hooks**: Analyse und Dokumentation
- **Session-Hooks**: Start, Ende und Wiederherstellung
- **Spezialisierte Hooks**: Notification, Memory-Update, Neural-Training [GitHub](#)

## Neural-Network-Parameter

Die V86-spezifischen neuronalen Features umfassen:

- **models**: Anzahl kognitiver Modelle (bis zu 27)
- **wasmsimd**: SIMD-Beschleunigung für Performance
- **training.patterns**: Lernmuster für Koordinationsverbesserung
- **training.epochs**: Trainingsiterationen
- **training.learningRate**: Lerngeschwindigkeit für Mustererkennung [npm](#)

## Zusätzliche Parameter und Dateien

Neben der Hauptkonfiguration werden weitere wichtige Dateien und Parameter übergeben, die das System-Verhalten beeinflussen.

## Umgebungsvariablen

```
bash
```



```
# Authentifizierung
```

```
ANTHROPIC_API_KEY=sk-ant-...
```

```
ANTHROPIC_BASE_URL=https://api.anthropic.com
```

```
ANTHROPIC_MODEL=claude-3-sonnet-20240229
```

```
# Claude-Flow spezifisch
```

```
CLAUDE_FLOW_HOOKS_ENABLED=true
```

```
CLAUDE_FLOW_TELEMETRY_ENABLED=true
```

```
CLAUDE_FLOW_DEBUG=verbose
```

```
CLAUDE_FLOW_MAX_AGENTS=12
```

```
CLAUDE_FLOW_MEMORY_PATH=.swarm/memory.db
```

## Begleitende Dateien

- **CLAUDE.md**: Projektkontext und Instruktionen für Claude Code
- **.swarm/memory.db**: SQLite-Datenbank mit 12 spezialisierten Tabellen
- **agents/\*.json**: 64 individuelle Agentenkonfigurationen
- **workflows/**: Automatisierungs- und Deployment-Workflows
- **commands/**: Dokumentation für alle 87 MCP-Tools

## MCP-Tool-Integration

Das System bietet **87 spezialisierte MCP-Tools** organisiert in Kategorien: [GitHub](#) [npm](#)

- **Core Swarm Tools**: swarm\_init, agent\_spawn, task\_orchestrate
- **Memory Tools**: memory\_usage, neural\_train, session\_restore
- **GitHub Tools**: github\_swarm, pr\_enhance, repo\_analyze
- **Monitoring Tools**: benchmark\_run, swarm\_monitor, performance\_track [GitHub +2](#)

## Hook-Integration für Automatisierung

```
bash
```

```
# Vor Aufgabenbeginn
```

```
npx claude-flow@alpha hooks pre-task --description "task" --auto-spawn-agents true
```

```
# Nach Dateioperationen
```

```
npx claude-flow@alpha hooks post-edit --file "filepath" --memory-key "swarm/agent/step"
```

```
# Session-Management
```

```
npx claude-flow@alpha hooks session-restore --session-id "swarm-id" --load-memory true
```

## Dokumentation und Strukturspezifikationen

Die technische Dokumentation für `claude-flow@alpha` (V86) ist primär in öffentlichen Repositories verfügbar, während interne Dokumentation möglicherweise in anderen Systemen gespeichert ist.

## Verfügbare Dokumentationsquellen

- **GitHub Repository:** <https://github.com/ruvnet/claude-flow> mit umfassender Wiki-Dokumentation GitHub
- **NPM Package:** <https://www.npmjs.com/package/claude-flow> mit Versionsinformationen npm
- **Beispielkonfigurationen:** Im [examples/01-configurations/](#) Verzeichnis des Repositories GitHub
- **Command-Dokumentation:** Automatisch generiert im [.claude/commands/](#) Verzeichnis

## Schema-Validierung

Das System implementiert eingebaute Validierung für:

- **Typprüfung:** Automatische Validierung von Datentypen
- **Wertebereichsprüfung:** Limits für numerische Parameter
- **Pflichtfeldvalidierung:** Erforderliche Felder werden erzwungen
- **Formatcompliance:** JSON-Schema-konforme Strukturen
- **Agentenfähigkeitsabgleich:** Kompatibilitätsprüfung zwischen Agenten

## Best Practices für Konfiguration

Die optimale Nutzung erfordert schrittweisen Aufbau der Konfiguration. **Beginnen Sie mit minimaler Konfiguration** und erweitern Sie nach Bedarf. Nutzen Sie **3-5 Agenten für einfache Aufgaben** und **8-12 für komplexe Projekte**. npm GitHub Aktivieren Sie **persistente Memory** für Cross-Session-Workflows und **Hooks für automatisierte Koordination**. Die Konfiguration sollte immer die **Version explizit spezifizieren** für Kompatibilität. GitHub

## Performance-Charakteristiken

Version 86 zeigt beeindruckende Leistungsmetriken mit **84.8% Erfolgsrate bei SWE-Bench**, **2.8-4.4x schnellerer Ausführung** durch parallele Koordination und **32.3% Token-Reduktion** durch intelligente Optimierung. GitHub +2 Das System unterstützt bis zu **12 gleichzeitige Agenten** mit automatischer Topologie-Auswahl basierend auf Aufgabenkomplexität.

Die Plattform bietet Enterprise-Grade-Features wie Byzantine Fault Tolerance, kryptografische Signierung, Audit-Trails und Echtzeit-Monitoring. GitHub Mit 64 spezialisierten Agenten, 87 MCP-Tools und 27 neuronalen Modellen stellt `claude-flow@alpha` (V86) eine umfassende AI-Orchestrierungslösung für komplexe Entwicklungsprojekte dar. GitHub +4