Claude-Flow Konfigurations-Synchronisation Guide

🔍 Zu prüfende Konfigurationsdateien

1. `.claude/settings.json` (Claude Code Hauptkonfiguration)

Übernehmen Sie:

```json
{
  "model": "sonnet",  // Ihr bevorzugtes Modell
  "permissions": {
    "allow": [
      // Ihre erlaubten Befehle aus der aktuellen Konfiguration
      "Bash(mkdir:*)",
      "Bash(npm:*)",
      "Bash(python:*)",
      "Write",
      "Edit",
      "MultiEdit",
      "Read",
      "Search",
      "Grep"
    ],
    "deny": [
      // Sicherheitsbeschränkungen
      "Bash(rm -rf:*)",
      "Bash(sudo:*)"
    ]
  },
  "env": {
    // Ihre Umgebungsvariablen
    "ANTHROPIC_API_KEY": "sk-ant-...",  // WICHTIG: Ihr API Key
    "BASH_DEFAULT_TIMEOUT_MS": "300000"
  },
  "hooks": {
    // Ihre existierenden Hooks
  }
}
```

2. `.mcp.json` (MCP Server Konfiguration)

KRITISCH für MCP-Funktionalität:

```json
```

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-filesystem"],
      "env": {
        "FILESYSTEM_ROOT": "D:\\03_Git\\02_Python\\01_AI_Coding_Station"
      }
    },
    "github": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-github"],
      "env": {
        "GITHUB_PERSONAL_ACCESS_TOKEN": "ghp_..."  // Ihr GitHub Token
      }
    },
    "git": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-git"]
    },
    "memory": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-memory"]
    },
    "puppeteer": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-puppeteer"],
      "env": {
        "PUPPETEER_HEADLESS": "true"
      }
    },
    "sqlite": {
      "command": "npx",
      "args": ["-y", "@modelcontextprotocol/server-sqlite"],
      "env": {
        "SQLITE_DB_PATH": ".swarm/memory.db"
      }
    },
    "claude-flow": {
      "command": "npx",
      "args": ["-y", "claude-flow@alpha", "mcp", "start"],
      "env": {}
    },
    "python": {
      "command": "npx",
      "args": ["-y", "mcp-server-python"],
      "env": {
        "PYTHON_PATH": "/usr/bin/python3"
      }
    }
  }
}
```

3. Umgebungsvariablen (`.env` oder System)

Prüfen und übernehmen:

```bash
# API Keys (KRITISCH!)
ANTHROPIC_API_KEY=sk-ant-api03-...  # Ihr echter API Key
ANTHROPIC_BASE_URL=https://api.anthropic.com
ANTHROPIC_MODEL=claude-3-sonnet-20240229

# GitHub (falls verwendet)
GITHUB_PERSONAL_ACCESS_TOKEN=ghp_...
GITHUB_USERNAME=your-username

# OpenAI (falls für Vergleiche verwendet)
OPENAI_API_KEY=sk-...

# 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

# Python-Umgebung
PYTHON_PATH=/usr/bin/python3
PYTHONPATH=D:\\03_Git\\02_Python\\01_AI_Coding_Station
VIRTUAL_ENV=.venv

# WSL spezifisch
WSL_DISTRO_NAME=Ubuntu
WSL_INTEROP=/run/WSL/...
```

4. `.claude/agents/` (Existierende Custom Agents)

Prüfen Sie vorhandene Agent-Definitionen:

```bash
```

```
# Listen Sie existierende Agents
ls -la .claude/agents/

# Beispiel: python-specialist.md
```

Falls vorhanden, übernehmen Sie diese in Ihre Konfiguration:

```json
{
  "agents": {
    "custom": [
      {
        "name": "python-specialist",
        "path": ".claude/agents/python-specialist.md",
        "enabled": true
      }
    ]
  }
}
```

5. `CLAUDE.md` (Projektkontext)

Erstellen/Aktualisieren Sie diese Datei:

```markdown
```

```
# 01_AI_Coding_Station - Python Development

## Project Context
This is a Python development environment with AI-powered coding assistance.

## Technology Stack
- Python 3.11+
- Django/Flask/FastAPI
- PostgreSQL/SQLite
- Docker/Kubernetes
- pytest/unittest

## Project Structure
- `/src` - Source code
- `/tests` - Test files
- `/docs` - Documentation
- `/.claude` - Claude configurations
- `/.swarm` - Hive mind memory

## Active MCP Servers
- filesystem - File operations
- github - Repository management
- git - Version control
- memory - Persistent storage
- sqlite - Database operations
- python - Python execution

## Development Workflow
1. Requirements analysis
2. Architecture design
3. Implementation
4. Testing
5. Documentation
6. Review and deployment
```

6. `.claude/settings.local.json` (Lokale Überschreibungen)

Falls vorhanden, hat Priorität:

```
json
```

```
{
  "model": "opus",  // Überschreibt globale Einstellung
  "customApiKey": "sk-ant-...",  // Falls anders als global
  "projectSpecific": {
    "pythonVersion": "3.11",
    "linter": "ruff",
    "formatter": "black"
  }
}
```

📝 Komplette Synchronisations-Checkliste

Schritt 1: Sammeln Sie Informationen

bash

```bash
# 1. Prüfen Sie existierende Claude-Einstellungen
cat .claude/settings.json

# 2. Prüfen Sie MCP-Konfiguration
cat .mcp.json

# 3. Listen Sie Umgebungsvariablen
env | grep -E "(ANTHROPIC|CLAUDE|GITHUB|PYTHON)"

# 4. Prüfen Sie vorhandene Agents
ls -la .claude/agents/

# 5. Prüfen Sie Hook-Konfigurationen
cat .claude/hooks.json 2>/dev/null || echo "No hooks file"
```

Schritt 2: Aktualisieren Sie die Haupt-Config

json

```
{
  "name": "01_AI_Coding_Station - 🐍 Python Development",
  "version": "2.0.0-alpha.86",

  // KRITISCH: Übernehmen aus .env oder settings.json
  "authentication": {
    "anthropicApiKey": "${ANTHROPIC_API_KEY}",  // Aus Umgebungsvariable
    "githubToken": "${GITHUB_PERSONAL_ACCESS_TOKEN}",
    "openaiApiKey": "${OPENAI_API_KEY}"
  },

  // KRITISCH: Aus .mcp.json übernehmen
  "mcp": {
    "enabled": true,
    "servers": {
      "filesystem": {
        "enabled": true,
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-filesystem"],
        "env": {
          "FILESYSTEM_ROOT": "D:\\03_Git\\02_Python\\01_AI_Coding_Station"
        }
      },
      "github": {
        "enabled": true,
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-github"],
        "env": {
          "GITHUB_PERSONAL_ACCESS_TOKEN": "${GITHUB_PERSONAL_ACCESS_TOKEN}"
        }
      },
      "git": {
        "enabled": true,
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-git"]
      },
      "memory": {
        "enabled": true,
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-memory"]
      },
      "sqlite": {
        "enabled": true,
        "command": "npx",
        "args": ["-y", "@modelcontextprotocol/server-sqlite"],
        "env": {
```

```
        "SQLITE_DB_PATH": ".swarm/memory.db"
      }
    },
    "python": {
      "enabled": true,
      "command": "npx",
      "args": ["-y", "mcp-server-python"],
      "env": {
        "PYTHON_PATH": "/usr/bin/python3",
        "PYTHONPATH": "D:\\03_Git\\02_Python\\01_AI_Coding_Station"
      }
    },
    "claude-flow": {
      "enabled": true,
      "command": "npx",
      "args": ["-y", "claude-flow@alpha", "mcp", "start"]
    }
  }
},

// Aus .claude/settings.json übernehmen
"permissions": {
  "allow": [
    "Bash(mkdir:*)",
    "Bash(npm:*)",
    "Bash(python:*)",
    "Bash(pip:*)",
    "Bash(pytest:*)",
    "Bash(git:*)",
    "Write",
    "Edit",
    "MultiEdit",
    "Read",
    "Search",
    "Grep",
    "MCP(*)"  // Erlaubt alle MCP-Tools
  ],
  "deny": [
    "Bash(rm -rf /*)",
    "Bash(sudo:*)",
    "Bash(chmod 777:*)"
  ]
},

// Hooks aus .claude/settings.json
"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}\""
        }
      ]
    },
    {
      "matcher": "*.py",
      "hooks": [
        {
          "type": "command",
          "command": "black --line-length 100 \"{}\" && mypy --strict \"{}\""
        }
      ]
    }
  ]
},

// Python-spezifische Einstellungen
"python": {
  "version": "3.11",
  "virtualEnv": ".venv",
  "packageManager": "pip",
  "linter": "ruff",
  "formatter": "black",
  "typeChecker": "mypy",
  "testRunner": "pytest",
  "dependencies": {
    "production": "requirements.txt",
    "development": "requirements-dev.txt"
  }
  }
}
```

🚀 Aktivierung Script

sync-config.sh

```bash
#!/bin/bash

echo "🔄 Synchronizing Claude-Flow Configuration..."

# 1. Backup existing configs
mkdir -p .claude-flow/backups
cp .claude-flow/saved-configs/*.json .claude-flow/backups/ 2>/dev/null || true

# 2. Check for required files
MISSING_FILES=""
[ ! -f ".claude/settings.json" ] && MISSING_FILES="$MISSING_FILES .claude/settings.json"
[ ! -f ".mcp.json" ] && MISSING_FILES="$MISSING_FILES .mcp.json"
[ -z "$ANTHROPIC_API_KEY" ] && MISSING_FILES="$MISSING_FILES ANTHROPIC_API_KEY"

if [ -n "$MISSING_FILES" ]; then
    echo "⚠️  Missing required configurations:"
    echo "$MISSING_FILES"
    echo "Please create/set these before continuing."
    exit 1
fi

# 3. Verify MCP servers
echo "🔍 Checking MCP servers..."
npx claude-flow@alpha mcp list

# 4. Initialize MCP servers
echo "🚀 Starting MCP servers..."
npx claude-flow@alpha mcp init --all

# 5. Test configuration
echo "✅ Testing configuration..."
npx claude-flow@alpha config validate \
    --file .claude-flow/saved-configs/20250819_Python-Development-Complete.json

echo "✨ Configuration synchronized successfully!"
```

⚠️ Kritische Punkte

1. **API Key ist PFLICHT**: Ohne `ANTHROPIC_API_KEY` funktioniert nichts
2. **MCP Server müssen installiert sein**:
   ```bash
   npm install -g @modelcontextprotocol/server-filesystem
   npm install -g @modelcontextprotocol/server-github
   # etc.
   ```
3. **Pfade müssen angepasst werden**: Windows vs WSL Pfade beachten
4. **Tokens sicher speichern**: Niemals in Git committen!

Führen Sie diese Synchronisation durch, bevor Sie claude-flow starten!