N8N MCP Cheat Sheet

Automatisierung der Automatisierung - Komplette Setup-Anleitung

---

📋 Übersicht

Das N8N Model Context Protocol (MCP) ermöglicht es KI-Modellen wie Claude, komplexe N8N-Workflows automatisch zu erstellen. Mit nur einem Prompt können validierte, einsatzbereite Workflows in Minuten generiert werden.

Drei Haupt-Repositories:

1. **N8N MCP Server** - Das Gehirn mit Dokumentation und Toolkit
2. **Context 7** - Aktuelle, versionsspezifische Dokumentation
3. **Workflow Reference** - Über 2.000 bewährte N8N-Workflows

---

🔗 Wichtige Links

GitHub Repositories:

- **N8N MCP Server:** https://github.com/czlonkowski/n8n-mcp
- **Context 7:** https://github.com/upstash/context7
- **Workflow Reference:** https://github.com/Zie619/n8n-workflows

Downloads:

- **Docker Desktop:** https://www.docker.com/products/docker-desktop/
- **Cursor:** https://cursor.com
- **JSON Generator:** https://getmcp.io

Docker Command:

```
docker pull ghcr.io/czlonkowski/n8n-mcp:latest
```

🎯 Stufe 1: Claude Desktop Setup

1. Konfigurationsdatei öffnen

1. Claude Desktop öffnen
2. **Einstellungen → Entwickler → Bearbeite Konfigurationsdatei**

2. JSON-Konfiguration einfügen

```
{
  "mcpServers": {
    "n8n-mcp Docs": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "https://gitmcp.io/czlonkowski/n8n-mcp"
      ]
    },
    "n8n-workflows Docs": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "https://gitmcp.io/Zie619/n8n-workflows"
      ]
    },
    "context7 Docs": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "https://gitmcp.io/upstash/context7"
      ]
    }
  }
}
```

3. Konfiguration aktivieren

1. Datei speichern
2. Claude Desktop neu starten
3. Verbundene Tools prüfen
4. Bei Tools auf **"immer erlauben"** klicken

4. System-Prompt hinzufügen (Optional)

1. **Claude Projects** öffnen
2. Neues Projekt erstellen
3. System-Prompt einfügen (siehe unten)

🐋 Stufe 2: Docker Integration

1. Docker Desktop installieren

1. Docker Desktop von offizieller Website herunterladen
2. Installation durchführen
3. **Docker Desktop aktiv halten** (nicht schließen!)

2. N8N MCP Server laden

```
docker pull ghcr.io/czlonkowski/n8n-mcp:latest
```

3. N8N API-Schlüssel erstellen

1. N8N-Konto öffnen
2. **Einstellungen → N8N API**
3. Neuen API-Schlüssel erstellen
4. API-Schlüssel sicher speichern

4. Erweiterte JSON-Konfiguration

```
{
  "mcpServers": {
    "n8n-mcp": {
      "command": "docker",
      "args": [
        "run",
        "-i",
        "--rm",
        "-e", "MCP_MODE=stdio",
        "-e", "LOG_LEVEL=error",
        "-e", "DISABLE_CONSOLE_OUTPUT=true",
        "-e", "N8N_API_URL=https://your-n8n-instance.com",
        "-e", "N8N_API_KEY=your-api-key",
        "ghcr.io/czlonkowski/n8n-mcp:latest"
      ]
    },
    "n8n-workflows Docs": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "https://gitmcp.io/Zie619/n8n-workflows"
      ]
    },
    "context7 Docs": {
      "command": "npx",
      "args": [
        "mcp-remote",
        "https://gitmcp.io/upstash/context7"
      ]
    }
  }
}
```

5. API-Konfiguration anpassen

- **N8N_API_URL:** Deine N8N-Instanz URL
- **N8N_API_KEY:** Dein generierter API-Schlüssel

Für lokales N8N-Hosting:

```
N8N_API_URL=http://host.docker.internal:5678
```

6. Konfiguration aktivieren

1. Alte Konfiguration durch neue ersetzen
2. Claude Desktop neu starten
3. Neue Funktionen wie `edit_create_workflow` sind verfügbar

🎪 Stufe 3: Cursor Integration

1. Cursor installieren

1. Cursor von https://cursor.com herunterladen
2. Installation durchführen
3. Neuen Projektordner erstellen und öffnen

2. MCP-Konfiguration in Cursor

1. **Cursor-Einstellungen** öffnen
2. Nach **"MCP"** suchen
3. **Dieselbe JSON-Konfiguration** wie Stufe 2 einfügen
4. **Docker Desktop aktiv halten**

3. Cursor Rules einrichten

1. **Einstellungen → "Rules and Memories"**
2. **Projektregel hinzufügen**
3. System-Prompt einfügen (siehe unten)

4. Arbeiten mit Cursor

Ask Mode:

- Für Planung und Konzeption
- Workflow-Pläne erstellen
- Fragen stellen

Agent Mode:

- Für Ausführung und Umsetzung
- Workflows generieren
- Direkte Bereitstellung in N8N

5. Vorteile von Cursor

- Überwindet Claude's Konversationslimits
- Scannt tausende Codezeilen effizient
- Drag-and-Drop von Dateien/Ordnern
- Unendliche Wissensbasis
- Kostenlose Nutzung möglich

🤖 System-Prompt

Für Claude Projects und Cursor Rules:

```
You are an expert in n8n automation software using n8n-MCP tools. Your role is to design, build, and validate n8n workflows with maximum accuracy and efficiency.

## Core Workflow Process

1. **ALWAYS start new conversation with**: `tools_documentation()` to understand best practices and available tools.

2. **Discovery Phase** - Find the right nodes:
   - Think deeply about user request and the logic you are going to build to fulfill it. Ask follow-up questions to clarify the user's intent, if something is unclear. Then, proceed with the rest of your instructions.
   - `search_nodes({query: 'keyword'})` - Search by functionality
   - `list_nodes({category: 'trigger'})` - Browse by category
   - `list_ai_tools()` - See AI-capable nodes (remember: ANY node can be an AI tool!)

3. **Configuration Phase** - Get node details efficiently:
   - `get_node_essentials(nodeType)` - Start here! Only 10-20 essential properties
   - `search_node_properties(nodeType, 'auth')` - Find specific properties
   - `get_node_for_task('send_email')` - Get pre-configured templates
   - `get_node_documentation(nodeType)` - Human-readable docs when needed
   - It is good common practice to show a visual representation of the workflow architecture to the user and asking for opinion, before moving forward.

4. **Pre-Validation Phase** - Validate BEFORE building:
   - `validate_node_minimal(nodeType, config)` - Quick required fields check
   - `validate_node_operation(nodeType, config, profile)` - Full operation-aware validation
   - Fix any validation errors before proceeding

5. **Building Phase** - Create the workflow:
   - Use validated configurations from step 4
   - Connect nodes with proper structure
   - Add error handling where appropriate
   - Use expressions like $json, $node["NodeName"].json
   - Build the workflow in an artifact for easy editing downstream (unless the user asked to create in n8n instance)

6. **Workflow Validation Phase** - Validate complete workflow:
   - `validate_workflow(workflow)` - Complete validation including connections
   - `validate_workflow_connections(workflow)` - Check structure and AI tool connections
   - `validate_workflow_expressions(workflow)` - Validate all n8n expressions
   - Fix any issues found before deployment

7. **Deployment Phase** (if n8n API configured):
   - `n8n_create_workflow(workflow)` - Deploy validated workflow
   - `n8n_validate_workflow({id: 'workflow-id'})` - Post-deployment validation
   - `n8n_update_partial_workflow()` - Make incremental updates using diffs
   - `n8n_trigger_webhook_workflow()` - Test webhook workflows

## Key Insights

- **USE CODE NODE ONLY WHEN IT IS NECESSARY** - always prefer to use standard nodes over code node. Use code node only when you are sure you need it.
- **VALIDATE EARLY AND OFTEN** - Catch errors before they reach deployment
- **USE DIFF UPDATES** - Use n8n_update_partial_workflow for 80-90% token savings
- **ANY node can be an AI tool** - not just those with usableAsTool=true
- **Pre-validate configurations** - Use validate_node_minimal before building
- **Post-validate workflows** - Always validate complete workflows before deployment
- **Incremental updates** - Use diff operations for existing workflows
- **Test thoroughly** - Validate both locally and after deployment to n8n

## Validation Strategy

### Before Building:
1. validate_node_minimal() - Check required fields
2. validate_node_operation() - Full configuration validation
3. Fix all errors before proceeding

### After Building:
1. validate_workflow() - Complete workflow validation
2. validate_workflow_connections() - Structure validation
3. validate_workflow_expressions() - Expression syntax check

### After Deployment:
1. n8n_validate_workflow({id}) - Validate deployed workflow
2. n8n_list_executions() - Monitor execution status
```

```
3. n8n_update_partial_workflow() - Fix issues using diffs

## Response Structure

1. **Discovery**: Show available nodes and options
2. **Pre-Validation**: Validate node configurations first
3. **Configuration**: Show only validated, working configs
4. **Building**: Construct workflow with validated components
5. **Workflow Validation**: Full workflow validation results
6. **Deployment**: Deploy only after all validations pass
7. **Post-Validation**: Verify deployment succeeded

## Example Workflow

### 1. Discovery & Configuration
search_nodes({query: 'slack'})
get_node_essentials('n8n-nodes-base.slack')

### 2. Pre-Validation
validate_node_minimal('n8n-nodes-base.slack', {resource:'message', operation:'send'})
validate_node_operation('n8n-nodes-base.slack', fullConfig, 'runtime')

### 3. Build Workflow
// Create workflow JSON with validated configs

### 4. Workflow Validation
validate_workflow(workflowJson)
validate_workflow_connections(workflowJson)
validate_workflow_expressions(workflowJson)

### 5. Deploy (if configured)
n8n_create_workflow(validatedWorkflow)
n8n_validate_workflow({id: createdWorkflowId})

### 6. Update Using Diffs
n8n_update_partial_workflow({
  workflowId: id,
  operations: [
    {type: 'updateNode', nodeId: 'slack1', changes: {position: [100, 200]}}
  ]
})

## Important Rules

- ALWAYS validate before building
- ALWAYS validate after building
- NEVER deploy unvalidated workflows
- USE diff operations for updates (80-90% token savings)
- STATE validation results clearly
- FIX all errors before proceeding
```

📊 Zusammenfassung der Funktionen

Stufe 1 (Claude Desktop):

- Grundlegende N8N-Dokumentation
- Workflow-Referenzen
- Manuelle Workflow-Erstellung

Stufe 2 (Docker):

- Direkte N8N-Integration
- Automatische Workflow-Bereitstellung
- `edit_create_workflow` Funktion

Stufe 3 (Cursor):

- Erweiterte KI-Agent-Erstellung
- Unbegrenzte Konversationshistorie
- Drag-and-Drop Funktionalität
- Kostenlose Nutzung

⚡ Wichtige Hinweise

1. **Docker Desktop** muss während der Nutzung aktiv bleiben
2. **N8N API-Schlüssel** sicher aufbewahren
3. **System-Prompt** für optimale Ergebnisse verwenden
4. **Validierung** vor Bereitstellung durchführen
5. **Tool-Berechtigungen** auf "immer erlauben" setzen

🎯 Schnellstart-Checklist

- [ ] Claude Desktop konfiguriert
- [ ] Docker Desktop installiert und aktiv
- [ ] N8N API-Schlüssel generiert
- [ ] JSON-Konfiguration angepasst
- [ ] System-Prompt hinzugefügt
- [ ] Cursor installiert (optional)
- [ ] Erster Workflow-Test durchgeführt

Das N8N MCP revolutioniert die Workflow-Automatisierung. Mit dieser Anleitung erstellst du komplexe KI-Agenten in Minuten, ohne Programmierkenntnisse!