AIS+ AI Automation Society

Self-Healing n8n Workflows

Complete Setup Guide for Connecting n8n Cloud to Claude Code

Version 1.0 | January 2026

Table of Contents

Overview

What This System Does

When a workflow fails in your n8n Cloud instance, this system automatically:

1. Catches the failure using n8n's Error Trigger
2. Sends error details through ngrok to your local computer
3. Spawns Claude Code to analyze the broken workflow
4. Claude identifies the problem and fixes it
5. The fixed workflow is pushed back to n8n Cloud automatically

> **Result**
>
> Your workflows fix themselves, even when you're not at your computer.

Why This is Useful

- **Automatic bug fixes** - Common errors like null reference errors, syntax issues, and expression problems get fixed without your intervention
- **24/7 availability** - As long as your local machine is running, Claude is ready to fix errors
- **Learning tool** - Watch Claude's fixes to learn better n8n patterns
- **Time saver** - Stop debugging simple workflow errors manually

What Claude Can and Cannot Fix

Claude CAN Fix	Claude CANNOT Fix
Null/undefined reference errors	Expired credentials (OAuth tokens)

Expression syntax errors	API outages (Slack/Google down)
Code node JavaScript bugs	Rate limiting (429 errors)
Data type mismatches	Account/billing issues
Missing null checks	Network/connectivity problems

Prerequisites

Before starting, make sure you have:

Requirement	Description	How to Get It
n8n Cloud Account	A running n8n instance	Sign up at n8n.io
n8n API Key	For Claude to access your instance	n8n Settings → API → Create Key
Node.js	Version 18 or higher	nodejs.org
Claude Code CLI	Anthropic's CLI tool	See Part 1 below
ngrok Account	Free account for tunneling	ngrok.com
Anthropic API Key	For Claude Code	console.anthropic.com

Architecture

Here's how all the pieces connect:

```
┌──────────────────────────────────────────────────────────┐
```

```
│                         INTERNET                          │
├──────────────────────────────────────────────────────────┤
│                                                          │
│   ┌──────────────┐               ┌──────────────┐        │
│   │  n8n Cloud   │               │    ngrok     │        │
│   │              │  HTTPS POST   │              │        │
│   │ Error Trigger ─┼──────────────►│Tunnel Service│        │
│   │       │      │               │      │       │        │
│   │       ▼      │               └──────┼───────┘        │
│   │ HTTP Request │                      │                │
│   │ (to ngrok URL)│                     │                │
│   └──────────────┘                      │                │
│                                         │                │
└─────────────────────────────────────────┼────────────────┘
                                          │
┌─────────────────────────────────────────┼────────────────┐
│                 YOUR LOCAL COMPUTER     │                │
├─────────────────────────────────────────┼────────────────┤
│                                         │                │
│   ┌──────────────┐        ┌─────────────▼─────────────┐  │
│   │  Claude Code │◄───────│      Bridge Server        │  │
│   │              │ spawns │     (localhost:3456)      │  │
│   │  ┌─────────┐ │        │                           │  │
│   │  │ n8n MCP │ │        │  • Receives error webhook │  │
│   │  │ Server  │ │        │  • Validates API key      │  │
│   │  └─────────┘ │        │  • Spawns Claude Code     │  │
│   │              │        │  • Returns fix results    │  │
│   │ Analyzes error,│      └───────────────────────────┘  │
│   │ fetches workflow,│                                   │
│   │ applies fix  │─────────────────────────────────────► │
│   │              │    Pushes fix to n8n Cloud via API    │
│   └──────────────┘                                       │
│                                                          │
└──────────────────────────────────────────────────────────┘
```

Part 1: Install Claude Code

Claude Code is Anthropic's command-line interface that allows Claude to execute commands and interact with external tools.

1 Install Claude Code

Open your terminal (PowerShell on Windows, Terminal on Mac/Linux) and run:

```
npm install -g @anthropic-ai/claude-code
```

2 Authenticate Claude Code

Run Claude Code for the first time:

```
claude
```

Follow the prompts to link your Anthropic account. You'll need an Anthropic API key with credits.

3 Verify Installation

```
claude --version
```

You should see the version number printed.

Part 2: Set Up the n8n MCP Server

The MCP server gives Claude the ability to interact with your n8n instance - fetching workflows, updating them, and validating changes.

1 Get Your n8n API Key

1. Log into your n8n Cloud instance
2. Click on your **profile icon** (bottom left)
3. Go to **Settings**
4. Click **API** in the left sidebar
5. Click **Create API Key**
6. Give it a name like "Claude Code"
7. Copy the API key

2 Note Your n8n Instance URL

Your n8n Cloud URL looks like: `https://your-instance.app.n8n.cloud`

3 Configure Claude Code

On Windows, create or edit this file:

```
C:\Users\YourUsername\AppData\Roaming\Claude\claude_desktop_config.json
```

On Mac/Linux, the file is at:

```
~/.config/claude/claude_desktop_config.json
```

Add this configuration:

```
{
  "mcpServers": {
    "n8n-mcp": {
      "command": "npx",
      "args": ["-y", "@czlonkowski/n8n-mcp@latest"],
      "env": {
        "N8N_HOST": "https://your-instance.app.n8n.cloud",
        "N8N_API_KEY": "your-n8n-api-key-here"
      }
    }
  }
}
```

> **Important**
>
> Replace `your-instance.app.n8n.cloud` with your actual n8n URL and `your-n8n-api-key-here` with your API key.

4 Test the MCP Connection

Open a new terminal, run `claude` , then ask:

```
List my n8n workflows
```

If configured correctly, Claude will fetch and display your workflows.

Part 3: Create the Bridge Server

The bridge server listens for errors from n8n and spawns Claude Code to fix them.

1 Create the Project Directory

```
mkdir claude-n8n-bridge
cd claude-n8n-bridge
```

2 Initialize the Project

```
npm init -y
npm install express
```

3 Create server.js

Create a file named `server.js` with the following content:

```
const express = require('express');
const { spawn } = require('child_process');
const fs = require('fs');
const path = require('path');
const app = express();

app.use(express.json());

const API_KEY = process.env.BRIDGE_API_KEY || 'change-me-to-something-secret';

// Health check endpoint
```

```
app.get('/health', (req, res) => {
  res.json({ status: 'ok', timestamp: new Date().toISOString() });
});

// Main endpoint - n8n calls this when a workflow fails
app.post('/fix-workflow', async (req, res) => {
  if (req.headers['x-api-key'] !== API_KEY) {
    return res.status(401).json({ error: 'Invalid API key' });
  }

  const { workflow, execution, trigger } = req.body;

  if (!workflow) {
    return res.status(400).json({ error: 'Missing workflow data' });
  }

  const errorMessage = execution?.error?.message ||
                       trigger?.error?.message || 'Unknown error';
  const lastNode = execution?.lastNodeExecuted || 'Unknown';
  const executionId = execution?.id || 'N/A';

  const prompt = `An n8n workflow has failed and needs to be fixed.

WORKFLOW INFO:
- Workflow ID: ${workflow.id}
- Workflow Name: ${workflow.name}
- Execution ID: ${executionId}
- Failed Node: ${lastNode}
- Error Message: ${errorMessage}

YOUR TASK:
1. Use n8n_get_workflow with id "${workflow.id}" to fetch the workflow
2. Analyze what went wrong
3. Use n8n_update_partial_workflow to apply the fix
4. Explain what you fixed and why`;

  console.log(`\nError received: ${workflow.name}`);
  console.log(`Failed node: ${lastNode}`);
  console.log(`Error: ${errorMessage}\n`);

  try {
    const result = await runClaudeCode(prompt);
    res.json({ success: true, workflowId: workflow.id });
  } catch (error) {
    res.status(500).json({ success: false, error: error.message });
  }
});

function runClaudeCode(prompt) {
  return new Promise((resolve, reject) => {
    const tempFile = path.join(__dirname, 'temp-prompt.txt');
    fs.writeFileSync(tempFile, prompt, 'utf8');

    // Windows: use powershell
    const claude = spawn('powershell', [
      '-Command',
      `Get-Content "${tempFile}" | claude --dangerously-skip-permissions`
    ], { timeout: 600000, cwd: __dirname });

    let output = '';

    claude.stdout.on('data', (data) => {
      output += data.toString();
      process.stdout.write(data.toString());
    });

    claude.stderr.on('data', (data) => {
      process.stderr.write(data.toString());
    });

    claude.on('close', (code) => {
      try { fs.unlinkSync(tempFile); } catch (e) {}
      code === 0 ? resolve(output) : reject(new Error(`Exit code ${code}`));
    });

    claude.on('error', (err) => {
      try { fs.unlinkSync(tempFile); } catch (e) {}
      reject(err);
    });
  });
}

const PORT = process.env.PORT || 3456;
app.listen(PORT, () => {
  console.log(`Bridge server running on http://localhost:${PORT}`);
  console.log(`Waiting for n8n errors...`);
});
```

> **Mac/Linux Users**
>
> Replace the `spawn('powershell', ...)` section with:
>
> `spawn('bash', ['-c', `cat "${tempFile}" | claude --dangerously-skip-permissions`])`

4 Create CLAUDE.md

Create a file named `CLAUDE.md` in the same directory. This gives Claude context about its task:

```
# n8n Workflow Auto-Fixer

You are an autonomous n8n workflow repair agent.

## Your Mission
1. Fetch the workflow to understand its structure
2. Analyze the error to identify the root cause
3. Apply the fix using the MCP tools
4. Verify the fix is correct

**Apply fixes directly without asking for permission.**

## Tools to Use
- n8n_get_workflow - ALWAYS first, fetch the workflow
- n8n_update_partial_workflow - Apply targeted fixes
- n8n_validate_workflow - Verify the fix

## Common Fixes
- "Cannot read properties of undefined" → Add optional chaining (?.)
- "is not a function" → Fix method name
- "Unexpected token" → Fix syntax errors
- Expression errors → Fix n8n expression syntax
```

5 Start the Bridge Server

Windows (PowerShell):

```
$env:BRIDGE_API_KEY="my-secret-key-12345"
node server.js
```

Mac/Linux:

```
export BRIDGE_API_KEY="my-secret-key-12345"
node server.js
```

Part 4: Expose with ngrok

ngrok creates a secure tunnel from the internet to your local machine.

1 Install ngrok

1. Go to **ngrok.com** and create a free account
2. Download ngrok for your operating system
3. Extract the executable to a known location

2 Authenticate ngrok

Get your auth token from the ngrok dashboard, then run:

```
ngrok config add-authtoken YOUR_AUTH_TOKEN_HERE
```

3 Start the Tunnel

Open a **new terminal** (keep the bridge server running) and run:

```
ngrok http 3456
```

You'll see output like:

```
Forwarding   https://abc123xyz.ngrok-free.app -> http://localhost:3456
```

Copy the https URL - you'll need this for n8n.

4 Test the Tunnel

Open a browser and visit:

```
https://your-ngrok-url.ngrok-free.app/health
```

You should see: `{"status":"ok","timestamp":"..."}`

> **Free ngrok URLs Change**
>
> The free ngrok URL changes every time you restart ngrok. For a permanent URL, you'll need a paid ngrok plan.

Part 5: Create the Error Handler Workflow

This n8n workflow catches errors and sends them to your bridge server.

1 Create a New Workflow

1. Log into your n8n Cloud instance
2. Click **Add Workflow**
3. Name it: `Error Handler - Claude Code Fixer`

2 Add the Error Trigger Node

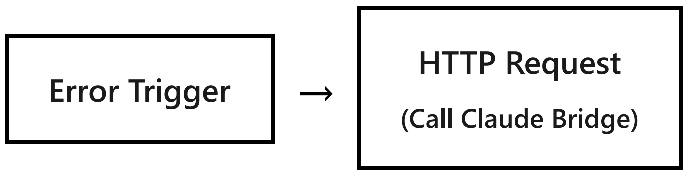
1. Click **Add first step**
2. Search for **Error Trigger**
3. Select it

3 Add the HTTP Request Node

Click the + button after Error Trigger, search for HTTP Request, and configure:

Setting	Value
Method	`POST`
URL	`https://your-ngrok-url.ngrok-free.app/fix-workflow`
Send Headers	ON
Header 1 Name	`x-api-key`
Header 1 Value	`my-secret-key-12345`
Header 2 Name	`ngrok-skip-browser-warning`
Header 2 Value	`true`
Send Body	ON
Body Content Type	`JSON`
Specify Body	`Using JSON`
JSON	`={{ JSON.stringify($json) }}`
Options → Timeout	`300000` (5 minutes)

4 Your Workflow Should Look Like This



5 Save and Activate

1. Click **Save** in the top right
2. Toggle the workflow to **Active**

Part 6: Connect Your Workflows

For Claude to fix a workflow when it fails, that workflow must be configured to use your error handler.

1 Open a Workflow You Want to Protect

Go to your n8n workflow list and open the workflow you want Claude to auto-fix.

2 Configure the Error Workflow Setting

1. Click the **three dots menu** (⋮) in the top right
2. Select **Settings**
3. Find **Error Workflow**
4. Select **Error Handler - Claude Code Fixer**
5. Click **Save**

3 Repeat for Other Workflows

Repeat for each workflow you want to protect with auto-fixing.

> **Tip**
>
> Start with a few test workflows to see how it works before connecting critical workflows.

Part 7: Test the System

Let's create a workflow that intentionally fails to test everything works.

1 Create a Test Workflow

1. Create a new workflow named: `TEST - Intentional Bug`
2. Add a **Manual Trigger** node
3. Add a **Code** node with this JavaScript:

```
// This code has an intentional bug - it will crash
const data = $input.all();

// Bug: accessing property on undefined
const userName = data[0].json.user.profile.name;

return [{ json: { name: userName } }];
```

4. Connect: Manual Trigger → Code

2 Connect to Error Handler

1. Click **Settings** (three dots menu)
2. Set **Error Workflow** to your error handler
3. Save

3 Run the Test

1. Make sure your bridge server is running
2. Make sure ngrok is running
3. In n8n, click **Test Workflow**
4. Watch your bridge server terminal

You should see:

- The workflow fails (expected)
- Error Handler triggers

- Bridge server receives the webhook
- Claude Code analyzes and fixes the workflow

4 Verify the Fix

1. Go back to your test workflow in n8n
2. Refresh the page
3. Open the Code node
4. You should see Claude has added null checks:

```
// Fixed: Added optional chaining and null checks
const data = $input.all();

const userName = data[0]?.json?.user?.profile?.name ?? 'Unknown';

return [{ json: { name: userName } }];
```

Troubleshooting

Problem	Cause	Solution
`claude: command not found`	Claude Code not installed	Run `npm install -g @anthropic-ai/claude-code`
`Connection refused`	Bridge server not running	Start the server with `node server.js`
`401 Invalid API key`	Keys don't match	Ensure n8n header matches BRIDGE_API_KEY
ngrok URL changed	Free URLs are temporary	Update URL in n8n workflow

Claude times out	Fix taking too long	Increase timeout in server.js and n8n node
Error Handler doesn't trigger	Workflow not connected	Set Error Workflow in workflow settings

Quick Reference

Start Everything (3 Terminals)

Terminal 1 - Bridge Server

```
cd claude-n8n-bridge
$env:BRIDGE_API_KEY="my-secret-key-12345"   # Windows
# export BRIDGE_API_KEY="my-secret-key-12345"  # Mac/Linux
node server.js
```

Terminal 2 - ngrok

```
ngrok http 3456
```

Key URLs

Component	URL
Bridge Server (local)	`http://localhost:3456/fix-workflow`
Health Check (local)	`http://localhost:3456/health`
Bridge Server (public)	`https://[your-ngrok-url].ngrok-free.app/fix-workflow`

Key Files

File	Purpose
`claude-n8n-bridge/server.js`	Bridge server code
`claude-n8n-bridge/CLAUDE.md`	Instructions for Claude
`claude_desktop_config.json`	MCP server configuration

AIS+ | AI Automation Society

Self-Healing n8n Workflows

Built with Claude Code + n8n MCP Server

To save as PDF: Press Ctrl+P (or Cmd+P on Mac) → Select "Save as PDF"