Programmatic Workflow Automation: Integrating the Gemini CLI with the n8n REST API

Part 1: The AI-Driven Automation Control Plane: A New Paradigm

1.1 Executive Summary: Beyond the GUI

This report details a complete framework for moving beyond the visual, "low-code" n8n interface and into a "meta-automation" paradigm. This architecture enables programmatic, terminal-based control over n8n workflows, leveraging the Google Gemini command-line interface (CLI) as an intelligent "programmer" and the n8n REST API as the execution "engine". This approach transforms n8n from a tool to be "clicked" into a platform to be "programmed."
We will present two distinct architectural methods to achieve this, each with specific technical trade-offs:

1. **The "Scripted" Method:** A flexible, universal approach utilizing standard Unix tools (curl, |, cat) to orchestrate a command-line-driven process. This method uses the Gemini CLI to generate or modify n8n workflow JSON, which is then piped to the n8n REST API for creation or updating.
2. **The "Integrated" Method:** An advanced, context-aware solution that utilizes a Model Context Protocol (MCP) server. This method "teaches" the Gemini CLI to be natively "n8n-aware," enabling it to understand n8n-specific tools and execute complex workflow management tasks directly with natural language commands.

1.2 Conceptual Framework: The Read-Modify-Write Cycle

The n8n REST API's design for workflow updates mandates a specific, three-phase development loop. The API endpoint for updating a workflow (PUT /api/v1/workflows/{id}) is a *full overwrite* operation, not a partial PATCH. It is not possible to send a small request to "change one node's parameter." Any update, no matter how small, requires sending the *entire*, complete workflow JSON object.
This constraint defines the central "Read-Modify-Write" cycle, which is the foundational-programmatic pattern for all n8n workflow management:

1. **Read:** The current state of the workflow is fetched. A curl GET request is sent to the n8n API endpoint (e.g., /api/v1/workflows/{id}) to retrieve the complete workflow definition as a JSON object. This JSON is saved to a local file.
2. **Modify:** The local JSON file is used as context for the Google Gemini CLI. A natural language prompt (e.g., "add a new Slack node after the 'Start' node") is provided to the gemini command. Gemini's generative AI engine parses the entire JSON, performs the complex logical changes, and outputs a new, *complete* JSON object.
3. **Write:** The new, modified JSON object is sent back to the n8n API. A curl PUT request (to

update) or POST (to create) is made, using the new JSON as the request body. This overwrites the old workflow with the new, modified version.

1.3 Architectural Overview: The Three Core Components

This entire system functions as a distributed control plane composed of three distinct components:

- **The Intelligent Frontend (Gemini CLI):** This is the natural language interface. Its sole function is to translate ambiguous, high-level human intent ("Create a workflow...") into the structured, unambiguous, machine-readable JSON payload that the n8n API requires.
- **The Execution Backend (n8n REST API):** This is the engine and the single source of truth. It is the stateless, programmatic interface for creating, reading, updating, deleting, activating, and deactivating all automation assets (workflows) on the n8n instance.
- **The Payload (n8n Workflow JSON):** This is the "code." This structured JSON document *is* the workflow, serving as the portable, serializable definition of all nodes, their parameters, and the connections between them. This is the data that is passed between the "Frontend" and the "Backend."

Part 2: The Execution Engine: Mastering the n8n REST API

2.1 Authentication: The X-N8N-API-KEY

All interaction with the n8n REST API is authenticated via a single API key. This key must be sent as an HTTP header named X-N8N-API-KEY.
This key is generated from within the n8n application's graphical user interface (GUI) by navigating to Settings > n8n API and selecting "Create an API key". This feature may not be available on all plans, such as free trials.
Security Best Practice: For programmatic and terminal-based use, this API key must **never** be hardcoded in scripts or passed as a command-line argument, as this would expose it in shell history. The industry-standard practice is to store the key and the n8n instance URL as environment variables in your shell's configuration file (e.g., ~/.bashrc or ~/.zshrc).
Implementation:

1. Add the following lines to your ~/.bashrc or ~/.zshrc file:
   ```
   # n8n API Configuration
   export N8N_URL="httpsa://your-n8n-instance.com"
   export N8N_API_KEY="n8n-api-key.your-secret-api-key-here"
   ```
2. Reload your shell to apply the changes: source ~/.bashrc.

All curl commands in this report will assume these environment variables are set.

2.2 Core Workflow Management: API Endpoints and curl

The following table provides the complete "cookbook" for all essential workflow management actions via curl and the n8n REST API.
Table 1: n8n Workflow API Endpoint Reference

Action	Method	Endpoint	curl Example (using Environment Variables)	References
Create Workflow	POST	/api/v1/workflows	curl -X POST "$N8N_URL/api/v1/workflows" -H "accept: application/json" -H "X-N8N-API-KEY: $N8N_API_KEY" -H "Content-Type: application/json" -d @new_workflow.json	
Read (List) All	GET	/api/v1/workflows	curl -X GET "$N8N_URL/api/v1/workflows" -H "accept: application/json" -H "X-N8N-API-KEY: $N8N_API_KEY"	
Read (Specific)	GET	/api/v1/workflows/{id}	curl -X GET "$N8N_URL/api/v1/workflows/123" -H "accept: application/json" -H "X-N8N-API-KEY: $N8N_API_KEY"	
Update Workflow	PUT	/api/v1/workflows/{id}	curl -X PUT "$N8N_URL/api/v1/workflows/123" -H "accept: application/json" -H "X-N8N-API-KEY: $N8N_API_KEY" -H "Content-Type: application/json" -d @modified_workflow.json	
Activate Workflow	POST	/api/v1/workflows/{id}/activate	curl -X POST "$N8N_URL/api/v1/workflows/123/acti	

Action	Method	Endpoint	curl Example (using Environment Variables)	References
			vate" -H "accept: application/json" -H "X-N8N-API-KEY: $N8N_API_KEY"	
Deactivate Workflow	POST	/api/v1/workflows/{id}/deactivate	curl -X POST "$N8N_URL/api/v1/workflows/123/deactivate" -H "accept: application/json" -H "X-N8N-API-KEY: $N8N_API_KEY"	

A common pitfall, especially in CI/CD pipelines, is assuming a workflow is "live" immediately after creation. Creating or updating a workflow with POST or PUT *does not* automatically activate it. A separate, subsequent POST request to the /api/v1/workflows/{id}/activate endpoint is mandatory to make the workflow operational.

2.3 Differentiating n8n API vs. n8n CLI

A critical distinction must be made between the **n8n REST API** and the **n8n CLI**. The user query is solvable only with the REST API.

- **n8n REST API:** This is a public, network-accessible interface for programmatically performing tasks (like those in Table 1) on a remote or local n8n instance. This is the correct tool for this architecture.
- **n8n CLI:** This refers to the server-side command-line tool n8n (e.g., n8n execute, n8n update:workflow). This CLI operates directly on the n8n database of a *self-hosted* instance. It is an administrative tool for the server, not a remote client for programmatic control. It cannot be used to control a remote n8n Cloud instance.

Part 3: The Intelligent Interface: Configuring the Google Gemini CLI

3.1 Installation and Authentication

The Google Gemini CLI is a Node.js-based application, available via the npm package manager.

- **Installation:** npm install -g @google/gemini-cli@latest
- **Authentication (Interactive):** For general use, run gemini auth and follow the browser-based sign-in flow.
- **Authentication (Scripting):** For non-interactive use in scripts or CI/CD pipelines, set the

GEMINI_API_KEY environment variable. You can obtain a key from Google AI Studio.

3.2 Key Capabilities for JSON Manipulation

For this architecture, the Gemini CLI cannot be used in its default *interactive* (chat) mode. It must be controlled non-interactively using specific flags and techniques.

- **Non-Interactive Mode (-p):** The -p (or --prompt) flag is essential. It allows you to pass a single prompt, receive a single response, and exit. This makes the gemini command usable within a bash script or pipeline.
- **Forcing Structured Content (Prompt Engineering):** The Gemini API supports a "JSON Mode" to *force* the model's output to be valid JSON. While the CLI does not expose a simple flag for this, the same result can be reliably achieved through prompt engineering. The prompt *must* contain an unambiguous instruction, such as: "...Respond ONLY with the valid, complete JSON object and no other text or markdown."
- **Forcing a Structured Wrapper (--output-format json):** This is a *scripting* feature, distinct from the "JSON Mode" prompt. This flag instructs the *Gemini CLI itself* to wrap its *entire output* (which includes the model's response, safety data, etc.) in a predictable JSON structure. This is invaluable for parsing, as a script can use a tool like jq to reliably extract the model's text response from the wrapper.
- **Providing Context (Workflow JSON):** There are two primary methods to provide the n8n workflow JSON to Gemini for modification:
    1. **Piping via stdin:** This is the standard Unix philosophy. The output of one command (e.g., cat workflow.json or a curl GET) is piped (|) directly to the gemini command as standard input.
    2. **File-based Context (@):** This is a Gemini-native feature. The prompt string can reference a local file using the @ prefix (e.g., gemini -p "@workflow.json Modify this file..."). This loads the file's contents into the model's context. This method is often cleaner for scripts and easier to debug, as it doesn't rely on a single, long pipe.

3.3 Managing Context (Persistent Instructions): GEMINI.md

To avoid repeating complex instructions (e.g., "You are an n8n workflow specialist...") in every prompt, the Gemini CLI supports a GEMINI.md context file.

- By creating a GEMINI.md file in the project directory, you provide persistent "memory" or instructions that are automatically loaded every time gemini is run in that directory.
- For this use case, a GEMINI.md file could contain:n8n Workflow Specialist ContextYou are an expert n8n workflow developer. Your task is to modify n8n workflow JSON.
    - You will be given a workflow JSON object and a natural language prompt.
    - Your response MUST be ONLY the new, complete, valid n8n workflow JSON.
    - Do not include any conversational text, markdown, or apologies.
    - Ensure all node IDs are unique and all connections are valid.

Part 4: The Payload: Understanding the n8n Workflow JSON Structure

To programmatically modify a workflow, the AI must understand its data structure. An n8n workflow *definition* (the file we are editing) is a single JSON object. This is separate from the

data that flows through a workflow, which is an array of objects.
The workflow JSON object has four key top-level properties :

1. name: The human-readable workflow name (e.g., "My Dynamic Workflow").
2. nodes: An array of objects, where each object defines a single node (a trigger, an action, etc.).
3. connections: An object defining the "wires" that link the nodes together.
4. settings: An object for workflow-level settings like timezone, execution order, and error workflow.

4.2 Anatomy of a Node Object

Each object inside the nodes array has a consistent structure:

```
{
  "id": "0f5532f9-36ba-4bef-86c7-30d607400b15",
  "name": "Jira",
  "type": "n8n-nodes-base.jira",
  "typeVersion": 1,
  "position": [-100, 80],
  "parameters": {
    "additionalProperties": {}
  },
  "credentials": {
    "jiraSoftwareCloudApi": {
      "id": "35",
      "name": "jiraApi"
    }
  }
}
```

Key fields to modify:

- id: A unique identifier for the node.
- name: The display name in the GUI.
- type: The node's "class" (e.g., n8n-nodes-base.set).
- parameters: This object contains all the user-configured settings for that node. This is where Gemini will make most of its changes.

4.3 Anatomy of the Connections Object

The connections object defines the "wires." This is the most complex part of a modification. The keys of this object are the *name* property of the *source* nodes (e.t., "Start"), not their id.
This structure means that a simple prompt like "add a Set node" is a sophisticated reasoning task. The AI cannot simply add a new object to the nodes array. It must also:

1. Generate a unique id for the new node.
2. Create the full parameters object for the new node.
3. Modify the connections object to "re-wire" the workflow, changing the target of the preceding node to point to the new node's name.

This complexity—especially the use of a fragile node name as a key—is precisely why a simple script fails and an LLM like Gemini, which can understand the *intent* and the *entire structur*e, is

the superior tool for the job.

Part 5: Method 1 (The "Scripted" Method): A Complete Read-Modify-Write Workflow

This section provides complete, production-ready bash scripts for managing n8n workflows from the terminal.

5.1 Preamble: Setting the Secure Environment

The following scripts are designed to be run in a terminal where the N8N_URL and N8N_API_KEY environment variables have been set, as described in Part 2.1.

5.2 Scenario A: Creating a New Workflow from a Natural Language Prompt

This script uses Gemini to generate a workflow from scratch and immediately uploads it to n8n.

```
#!/bin/bash
# create_workflow.sh

# 1. DEFINE THE PROMPT
# The prompt is highly specific to ensure we get ONLY raw JSON back.
PROMPT="Generate a complete n8n workflow JSON. The workflow must:
1. Trigger with a Webhook node named 'Start'.
2. Connect to a Set node named 'Set Message' that sets a variable
'message' to 'Hello from Gemini'.
3. Connect 'Set Message' to a 'Respond to Webhook' node.
4. Respond ONLY with the raw, valid JSON. Do not include markdown or
any other text."

echo "Generating workflow JSON from Gemini..."

# 2. GENERATE
# We use 'gemini -p' for non-interactive mode.
# We redirect output to a file for inspection and for the 'curl -d @'
command.
gemini -p "$PROMPT" > new_workflow.json

if [! -s new_workflow.json ]; then
    echo "Error: Gemini did not produce an output file. Check Gemini
auth and prompt."
    exit 1
fi

echo "Uploading new workflow to n8n..."
```

```
# 3. WRITE
# -s for silent, -w "\n%{http_code}" to print the HTTP code on a new
line.
RESPONSE=$(curl -s -w "\n%{http_code}" -X POST
"$N8N_URL/api/v1/workflows" \
    -H "accept: application/json" \
    -H "X-N8N-API-KEY: $N8N_API_KEY" \
    -H "Content-Type: application/json" \
    -d @new_workflow.json)

# 4. PARSE RESPONSE
HTTP_CODE=$(echo "$RESPONSE" | tail -n1)
BODY=$(echo "$RESPONSE" | sed '$d')

if; then
    echo "Successfully created workflow."
    echo "$BODY"

    # Optional: Automatically activate the new workflow
    # WORKFLOW_ID=$(echo "$BODY" | grep -o '"id":"[^"]*' | cut -d'"'
-f4)
    # if; then
    #    echo "Activating workflow $WORKFLOW_ID..."
    #    curl -X POST
"$N8N_URL/api/v1/workflows/$WORKFLOW_ID/activate" \
    #         -H "X-N8N-API-KEY: $N8N_API_KEY"
    # fi
else
    echo "Failed to create workflow. HTTP Code: $HTTP_CODE"
    echo "Response: $BODY"
fi
```

This script relies on Gemini's general knowledge of n8n, which is "context-blind". It works well for common nodes but may fail with complex or custom nodes.

5.3 Scenario B: The Full Read-Modify-Write Cycle (Update)

This script performs the complete cycle: fetching an existing workflow, using Gemini to modify it, and uploading the new version.

```
#!/bin/bash
# update_workflow.sh

if [ -z "$1" ]; then
    echo "Usage: $0 <WORKFLOW_ID>"
    exit 1
fi

WORKFLOW_ID=$1
```

```
PROMPT="Review the following n8n workflow JSON. Add a new 'Set' node
with the name 'Gemini-Modified' that sets a variable 'status' to
'updated'. Connect it between the 'Start' node and whatever 'Start'
connects to. Respond ONLY with the new, complete, valid JSON."

# 1. READ
echo "Fetching workflow $WORKFLOW_ID..."
RESPONSE=$(curl -s -w "\n%{http_code}" -X GET
"$N8N_URL/api/v1/workflows/$WORKFLOW_ID" \
    -H "accept: application/json" \
    -H "X-N8N-API-KEY: $N8N_API_KEY")

HTTP_CODE=$(echo "$RESPONSE" | tail -n1)
BODY=$(echo "$RESPONSE" | sed '$d')

if; then
    echo "Error fetching workflow. HTTP Code: $HTTP_CODE"
    echo "Response: $BODY"
    exit 1
fi

echo "$BODY" > current_workflow.json

# 2. MODIFY
echo "Modifying workflow with Gemini..."
# Here we use the @file context method
[span_74](start_span)[span_74](end_span)
gemini -p "@current_workflow.json $PROMPT" > modified_workflow.json

if [! -s modified_workflow.json ]; then
    echo "Error: Gemini failed to produce a modified workflow."
    exit 1
fi

# 3. WRITE
echo "Uploading modified workflow..."
RESPONSE=$(curl -s -w "\n%{http_code}" -X PUT
"$N8N_URL/api/v1/workflows/$WORKFLOW_ID" \
    -H "accept: application/json" \
    -H "X-N8N-API-KEY: $N8N_API_KEY" \
    -H "Content-Type: application/json" \
    -d @modified_workflow.json)

HTTP_CODE=$(echo "$RESPONSE" | tail -n1)
BODY=$(echo "$RESPONSE" | sed '$d')

if; then
    echo "Successfully updated workflow $WORKFLOW_ID."
```

```
    echo "$BODY"
else
    echo "Failed to update workflow. HTTP Code: $HTTP_CODE"
    echo "Response: $BODY"
fi
```

Part 6: Method 2 (The "Integrated" Method): The n8n Model Context Protocol (MCP) Server

6.1 The Paradigm Shift: From "Blind" to "Context-Aware"

The "Scripted" method in Part 5 is powerful but "context-blind". It relies on Gemini's general, pre-trained knowledge. A more robust, expert-level solution is to make the Gemini CLI "context-aware" by integrating it with a **Model Context Protocol (MCP) Server**.
An MCP server acts as a "Rosetta Stone" or "bridge," allowing an LLM (like Gemini) to discover and use external tools (like the n8n API) as native functions. The open-source n8n-mcp project is built for this exact purpose. It exposes your n8n instance's nodes, parameters, and documentation as a set of tools that the Gemini CLI can natively understand and call.

6.2 The n8n-mcp Project: What It Does

The n8n-mcp server provides :

- **Smart Node Discovery:** Allows Gemini to query for nodes (e.g., "what nodes exist for 'Google Sheets'?").
- **Workflow Validation:** Provides parameter schemas to the AI, dramatically reducing errors.
- **n8n Management Tools:** Exposes high-level functions like create_workflow and update_workflow, which the AI can call directly, abstracting away the need to manually generate JSON.

6.3 Step-by-Step Setup

1. **Install n8n-mcp:**
   ```
   git clone https://github.com/czlonkowski/n8n-mcp.git
   cd n8n-mcp
   npm install
   ```
   (Note: Alternate n8n-mcp-server projects also exist, e.g.).
2. **Configure n8n-mcp:** Create a configuration file (e.g., config.local.json) and add your n8n instance details :
   ```
   {
     "n8n": {
       "url": "httpsa://your-n8n-instance.com",
       "apiKey": "n8n-api-key.your-secret-api-key-here"
     }
   }
   ```

3. **Run the MCP Server:** npm start -- --config=config.local.json This will start the MCP server, typically on http://localhost:8080.
4. **Configure Gemini CLI to use the MCP Server:** Edit your Gemini CLI settings file (code ~/.gemini/settings.json). Add the local MCP server to the mcpServers array :
```
{
  ...
  "mcpServers": [
    "http://localhost:8080"
  ]
}
```
5. **Restart Gemini CLI:** Relaunch the gemini command. It will now automatically connect to your n8n-mcp server and load its "tools."

6.4 The New "Context-Aware" Workflow

With the n8n-mcp server running, the entire interaction model changes from "text manipulation" to "tool execution."

- **Old (Scripted) Method:** cat workflow.json | gemini "Modify this JSON to add a Slack node..."
- **New (Integrated) Method:** gemini "@n8n create_workflow 'Create a workflow that triggers on a webhook and sends a Slack message to #general'"

In this new model, Gemini CLI receives the prompt, consults its loaded MCP tools, and identifies the @n8n toolset. It understands that create_workflow is an available function. It then intelligently calls that function, and the n8n-mcp server handles the complex task of generating the valid JSON and calling the n8n REST API *on its own*. This is a far more robust, reliable, and powerful architecture, representing the true "end of drag-and-drop".

Part 7: Comparative Analysis and Strategic Recommendations

7.1 Analysis: GUI vs. Scripted API vs. Integrated MCP

The optimal method for n8n workflow management is not uniform; it is dependent on the specific task, technical environment, and desired reliability. The following table provides a decision matrix.

Table 2: Comparison of n8n Workflow Management Methods

Metric	n8n GUI	Scripted API (Part 5)	Integrated MCP API (Part 6)
Setup Complexity	None	**Low** (Requires curl, gemini)	**High** (Requires a persistent Node.js server)
Context-Awareness	N/A (Visual)	**None** (AI is "blind," guessing JSON)	**Full** (AI is "aware" of nodes/params)
Reliability	High (for manual edits)	**Low** (Brittle;	**Very High** (Robust;

Metric	n8n GUI	Scripted API (Part 5)	Integrated MCP API (Part 6)
		prompt-dependent)	tool-based)
Use Case: Simple Edit	**Fastest**	Slow (Full Read-Modify-Write)	Fast (Single command)
Use Case: Complex New Workflow	Slow (Manual drag-and-drop)	Slow (High-effort prompt engineering)	**Fastest** (AI performs generation)
Use Case: CI/CD & GitOps	Not possible	**Good** (for simple, stateless tasks)	Excellent (but adds setup complexity)
References			

7.2 Final Expert Recommendations

1. **For Prototyping, Debugging, and Visual Flow:** Use the **n8n GUI**. The visual interface is unparalleled for understanding the flow of data, debugging node-by-node executions, and rapidly prototyping new ideas.
2. **For CI/CD, GitOps, and Stateless Deployment:** Use the **"Scripted" API Method (Part 5)**. This architecture is ideal for DevOps environments. If the goal is to *deploy* or *activate* a workflow.json file that is already complete and stored in a Git repository, the curl-based scripts are lightweight, stateless, and perfect. This method is for *deploying* a known-good artifact, not *creating* a new one.
3. **For AI-Native Development and Programmatic Control:** Use the **"Integrated" MCP Method (Part 6)**. This is the definitive "power" solution and the most advanced implementation. If the strategic goal is to *use natural language as the primary development interface*—to have Gemini act as a true pair-programmer for automation—this is the only robust, scalable, and context-aware solution. The high initial setup cost is justified by the exponential increase in capability and reliability.

Works cited

1. n8n Review: Features, Pricing, Pros & Cons (2025) - Siit, https://www.siit.io/tools/trending/n8n-review 2. Gemini CLI | Gemini Code Assist - Google for Developers, https://developers.google.com/gemini-code-assist/docs/gemini-cli 3. n8n public REST API Documentation and Guides - n8n Docs, https://docs.n8n.io/api/ 4. Why I ditched Claude for Google Gemini (And how n8n-MCP made both 10x smarter), https://medium.com/mcp-server/why-i-ditched-claude-for-google-gemini-and-how-n8n-mcp-made-both-10x-smarter-c088a4d23ff2 5. n8n MCP - AI-Powered n8n Workflow Automation, https://www.n8n-mcp.com/ 6. Partial Workflow Update via API - Questions - n8n Community, https://community.n8n.io/t/partial-workflow-update-via-api/139289 7. Update a Workflow using n8n API - Questions, https://community.n8n.io/t/update-a-workflow-using-n8n-api/41899 8. I need a available node & node details for create workflow by using your rest api, https://community.n8n.io/t/i-need-a-available-node-node-details-for-create-workflow-by-using-your-rest-api/115909 9. Cheatsheet - Gemini CLI - HackingNote, https://www.hackingnote.com/en/cheatsheets/gemini/ 10. google-gemini/gemini-cli: An open-source AI agent that brings the power of Gemini directly into your terminal. - GitHub, https://github.com/google-gemini/gemini-cli 11. Gemini CLI is awesome! But only when you make Claude Code use ..., https://www.reddit.com/r/ChatGPTCoding/comments/1lm3fxq/gemini_cli_is_awesome_but_only

_when_you_make/ 12. Problem with N8N API Key - Questions - n8n Community, https://community.n8n.io/t/problem-with-n8n-api-key/188330 13. Structured output | Gemini API - Google AI for Developers, https://ai.google.dev/gemini-api/docs/structured-output 14. Generate structured output (like JSON and enums) using the Gemini API | Firebase AI Logic, https://firebase.google.com/docs/ai-logic/generate-structured-output 15. Authentication | n8n Docs, https://docs.n8n.io/api/authentication/ 16. Understanding the data structure - n8n Docs, https://docs.n8n.io/courses/level-two/chapter-1/ 17. Create Dynamic Workflows Programmatically via Webhooks & n8n API, https://n8n.io/workflows/4544-create-dynamic-workflows-programmatically-via-webhooks-and-n8n-api/ 18. How to use API key in cURL for authentication - Simplified Guide, https://www.simplified.guide/curl/authenticate-api-key 19. Best Practices for API Key Safety | OpenAI Help Center, https://help.openai.com/en/articles/5112595-best-practices-for-api-key-safety 20. Configuration methods - n8n Docs, https://docs.n8n.io/hosting/configuration/configuration-methods/ 21. Using Gemini API keys - Google AI for Developers, https://ai.google.dev/gemini-api/docs/api-key 22. How to include environment variable in bash line CURL? - Super User, https://superuser.com/questions/835587/how-to-include-environment-variable-in-bash-line-curl 23. Activate and deactivate workflows on schedule using native n8n API, https://n8n.io/workflows/3229-activate-and-deactivate-workflows-on-schedule-using-native-n8n-api/ 24. Schedule Activate and Deactivate Workflow - Feature Requests - n8n Community, https://community.n8n.io/t/schedule-activate-and-deactivate-workflow/2440 25. Activate workflow CLI command not activating web hooks - Questions - n8n Community, https://community.n8n.io/t/activate-workflow-cli-command-not-activating-web-hooks/40756 26. Activating a workflow from the API results in timeout and workflow staying Inactive · Issue #7258 · n8n-io/n8n - GitHub, https://github.com/n8n-io/n8n/issues/7258 27. Activate and deactivate workflows via another workflow or api - Questions - n8n Community, https://community.n8n.io/t/activate-and-deactivate-workflows-via-another-workflow-or-api/1957 28. CLI commands - n8n Docs, https://docs.n8n.io/hosting/cli-commands/ 29. N8n cli only package - Questions - n8n Community, https://community.n8n.io/t/n8n-cli-only-package/88959 30. Gemini CLI 3.0: NEW GenKit Extension! Powerful AI Coding Agent Beats Claude Code!, https://www.youtube.com/watch?v=-f3F-0djMgs 31. Say hello to a new level of interactivity in Gemini CLI - Google Developers Blog, https://developers.googleblog.com/en/say-hello-to-a-new-level-of-interactivity-in-gemini-cli/ 32. Supercharge Code Workflows with Gemini CLI + JSON Prompts Parameters - Medium, https://medium.com/@and.gpch.dev/supercharge-code-workflows-with-gemini-cli-json-prompts-parameters-5ce9deaf15b2 33. Google Gemini CLI Cheatsheet - Philschmid, https://www.philschmid.de/gemini-cli-cheatsheet 34. Hands-on with Gemini CLI - Google Codelabs, https://codelabs.developers.google.com/gemini-cli-hands-on 35. Gemini CLI Tutorial Series — Part 9: Understanding Context, Memory and Conversational Branching | by Romin Irani | Google Cloud - Medium, https://medium.com/google-cloud/gemini-cli-tutorial-series-part-9-understanding-context-memory-and-conversational-branching-095feb3e5a43 36. Gemini CLI Context Management: Complete Workflow Guide, https://artofcoding.dev/building-a-context-aware-gemini-cli-workflow 37. How to handle Settings in N8N API Update Workflow - Feature Requests - n8n Community, https://community.n8n.io/t/how-to-handle-settings-in-n8n-api-update-workflow/30019 38. BREAKING! Generate n8n workflow from a text prompt! - Tips ..., https://community.n8n.io/t/breaking-generate-n8n-workflow-from-a-text-prompt/80400 39. How to use n8n with MCP - Hostinger, https://www.hostinger.com/tutorials/how-to-use-n8n-with-mcp

40. Building a Custom MCP Server for Gemini CLI: A Hands-on guide, https://medium.com/google-cloud/building-a-custom-mcp-server-for-gemini-cli-a-personal-finance-assistant-tutorial-ee230229ab7d 41. czlonkowski/n8n-mcp: A MCP for Claude Desktop / Claude Code / Windsurf / Cursor to build n8n workflows for you - GitHub, https://github.com/czlonkowski/n8n-mcp 42. Power-Up Your Gemini CLI with Your Own FastMCP Server Extension, https://medium.com/google-cloud/power-up-your-gemini-cli-with-your-own-fastmcp-server-extension-75f1e78d0047 43. leonardsellem/n8n-mcp-server: MCP server that provides tools and resources for interacting with n8n API - GitHub, https://github.com/leonardsellem/n8n-mcp-server 44. How to Install n8n-mcp Server in Claude Desktop To Vibe Code Automation Workflows, https://www.youtube.com/watch?v=7Egx498mtj0 45. The End of Drag-and-Drop Generate n8n Workflows with ONE ..., https://websensepro.com/blog/the-end-of-drag-and-drop-generate-n8n-workflows-with-one-prompt-2/