Here are templates for the three main types of files you should include in your knowledge base:

## **1. n8n Cheat Sheet Guide**

markdown

# n8n Comprehensive Cheat Sheet Guide

## Core n8n Structure and Concepts

### Workflow JSON Structure

- n8n workflows are stored as JSON objects

- Each node has a unique ID, position coordinates, and configuration

- Connections between nodes are defined in the "connections" object

- The core structure follows this pattern:

```json

{

"nodes": [

{

"id": "uuid-here",

"name": "Node Name",

"type": "node.type",

"typeVersion": 1,

"position": [x, y],

"parameters": {

// Node-specific configuration

}

}

],

"connections": {

"Node Name": {

"main": [

[

{

"node": "Next Node Name",

"type": "main",

"index": 0

}

]

]

}

},

"settings": {},

"staticData": null

}

### **Essential Node Types**

#### **Trigger Nodes**

* Webhook: Receives HTTP requests (webhook)
* Cron: Runs at scheduled intervals (cron)
* Slack: Monitors Slack messages (slack)
* WhatsApp: Receives WhatsApp messages (whatsapp)
* Gmail: Monitors emails (gmail)

#### **Processing Nodes**

* Function: Custom JavaScript code (function)
* Set: Manipulates data (set)
* Split In Batches: Processes arrays in batches (splitInBatches)
* IF: Conditional logic (if)
* Switch: Multi-path conditional logic (switch)
* Merge: Combines data from multiple sources (merge)
* AI Agent: Processes with LLMs (ai-agent)

#### **Output Nodes**

* HTTP Request: Makes API calls (httpRequest)
* Slack: Sends Slack messages (slack)
* WhatsApp: Sends WhatsApp messages (whatsApp)
* Google Docs: Creates or modifies docs (googleDocs)
* Email: Sends emails (email)

### **Common Data Patterns**

#### **Accessing Previous Node Data**

* Use $json to access data from previous nodes
* Example: {{$json.data.message}}

#### **Setting Variables**

json

{

"parameters": {

"values": {

"string": [

{

"name": "variableName",

"value": "{{$json.someValue}}"

}

]

}

}

}

## **Node-Specific Configuration**

### **AI Agent Node**

json

{

"parameters": {

"mode": "completion",

"aiProvider": "openai",

"authentication": "predefinedCredentialType",

"resource": "chatCompletion",

"options": {

"model": "gpt-4o",

"prompt": {

"content": "Your prompt here\n\nContext: {{$json.context}}",

"role": "user"

},

"temperature": 0.7,

"maxTokens": 1000

}

}

}

### **Webhook Trigger**

json

{

"parameters": {

"path": "custom-webhook-path",

"responseMode": "lastNode",

"options": {}

}

}

### **WhatsApp Trigger**

json

{

"parameters": {

"event": "message",

"source": "all",

"conditions": {

"string": [

{

"value1": "{{$json.type}}",

"operation": "equal",

"value2": "text"

}

]

}

}

}

### **Function Node (for custom logic)**

json

{

"parameters": {

"functionCode": "// Code here\nconst items = $input.all();\nconst newItems = items.map(item => {\n // Process item\n return item;\n});\nreturn newItems;"

}

}

### **Google Docs Node**

json

{

"parameters": {

"resource": "document",

"operation": "create",

"documentId": "",

"options": {

"title": "{{$json.title}}",

"content": "{{$json.content}}"

}

}

}

### **Sticky Notes (for documentation)**

json

{

"name": "Sticky Note",

"type": "n8n-nodes-base.stickyNote",

"typeVersion": 1,

"position": [x, y],

"parameters": {

"text": "This node does X and Y",

"height": 200,

"width": 400

}

}

## **Common Workflow Patterns**

### **Data Transformation Flow**

1. Trigger Node → Get input data
2. Function Node → Transform data with custom code
3. Set Node → Format data for next step
4. Output Node → Send processed data

### **Conditional Branching**

1. Trigger Node → Get input data
2. IF Node → Check condition
3. Path A → Handle one case
4. Path B → Handle alternative case
5. Merge Node → Combine results if needed

### **AI Processing Pipeline**

1. Trigger Node → Get input text/data
2. Function Node → Prepare context/prompt
3. AI Agent Node → Process with LLM
4. Set Node → Format response
5. Output Node → Deliver results

## 2. n8n Tips and Tricks

```markdown

# n8n Tips and Tricks

## General n8n Node Structure

- Every node needs a unique ID, usually a UUID

- Position coordinates are [x, y] where x increases to the right, y increases downward

- Nodes connect via the connections object, always pointing from one node to the next

## AI Agent Module

The AI Agent node is a powerful way to integrate LLMs like GPT-4 into your workflows.

### AI Agent Configuration

```json

{

"name": "AI Agent",

"type": "n8n-nodes-base.ai-agent",

"typeVersion": 1,

"position": [

980,

460

],

"parameters": {

"mode": "completion",

"aiProvider": "openai",

"authentication": "predefinedCredentialType",

"resource": "chatCompletion",

"options": {

"model": "gpt-4o",

"messages": {

"values": [

{

"content": "You are a helpful assistant. {{$json.userMessage}}",

"role": "user"

}

]

},

"temperature": 0.7,

"maxTokens": 1000

}

}

}

### **Available AI Tools**

* OpenAI: ChatGPT-4o (use "gpt-4o" as model ID)
* Anthropic: Claude (if credential configured)
* Hugging Face: Various models
* Custom API: For other LLM providers

## **Sticky Note Creation**

Sticky notes help document your workflow:

json

{

"name": "Sticky Note",

"type": "n8n-nodes-base.stickyNote",

"typeVersion": 1,

"position": [

840,

300

],

"parameters": {

"text": "This node processes the incoming message with an LLM",

"height": 200,

"width": 400,

"color": "yellow"

}

}

## **Common Tools Configuration**

### **WhatsApp**

json

{

"name": "WhatsApp Trigger",

"type": "n8n-nodes-base.whatsAppTrigger",

"position": [

300,

300

],

"parameters": {

"event": "message",

"source": "all"

}

}

### **Google Docs**

json

{

"name": "Google Docs",

"type": "n8n-nodes-base.googleDocs",

"position": [

700,

300

],

"parameters": {

"resource": "document",

"operation": "create",

"options": {

"title": "{{$json.documentTitle}}",

"content": "{{$json.documentContent}}"

}

}

}

### **Slack**

json

{

"name": "Slack",

"type": "n8n-nodes-base.slack",

"position": [

900,

300

],

"parameters": {

"resource": "message",

"channel": "{{$json.channelName}}",

"text": "{{$json.messageText}}"

}

}

### **Pinecone**

For vector database operations:

json

{

"name": "Pinecone",

"type": "n8n-nodes-base.pinecone",

"position": [

800,

400

],

"parameters": {

"operation": "upsert",

"index": "your-index-name",

"vectors": "{{$json.vectors}}",

"namespace": "{{$json.namespace}}"

}

}

### **Airtable**

json

{

"name": "Airtable",

"type": "n8n-nodes-base.airtable",

"position": [

600,

400

],

"parameters": {

"application": "your-base-id",

"table": "your-table-name",

"operation": "append",

"options": {

"fields": {

"field1": "{{$json.value1}}",

"field2": "{{$json.value2}}"

}

}

}

}

## **Common Error Handling Patterns**

* Use Error Trigger nodes to catch errors in main workflow
* Use IF nodes with error checking conditions
* Add Set nodes to log error details
* Use multiple branches to create fallback paths

## **Advanced Tips**

* For complex data transformations, use Function nodes with JavaScript
* For scheduling, use Cron nodes with proper cron syntax
* For pagination, use Loop nodes to handle multiple API pages
* Connect AI Agent nodes to multiple tools for agentic workflows
* Use n8n expressions like {{$json.field}} to access data dynamically

## 3. Example Workflow JSON

Here's a simple example workflow using an AI agent that you can include in your knowledge base:

```json

{

"nodes": [

{

"parameters": {

"triggerOn": "all",

"options": {}

},

"id": "1a2b3c4d-5e6f-7g8h-9i0j-1k2l3m4n5o6p",

"name": "Webhook",

"type": "n8n-nodes-base.webhook",

"typeVersion": 1,

"position": [

580,

300

]

},

{

"parameters": {

"mode": "completion",

"aiProvider": "openai",

"authentication": "predefinedCredentialType",

"resource": "chatCompletion",

"options": {

"model": "gpt-4o",

"messages": {

"values": [

{

"content": "You are a helpful AI assistant. Please respond to the following question in a helpful, accurate, and concise way:\n\n{{$json.body.question}}",

"role": "user"

}

]

},

"temperature": 0.7,

"maxTokens": 2000,

"additionalProperties": {}

}

},

"id": "2b3c4d5e-6f7g-8h9i-0j1k-2l3m4n5o6p7q",

"name": "AI Agent",

"type": "n8n-nodes-base.ai-agent",

"typeVersion": 1,

"position": [

780,

300

]

},

{

"parameters": {

"options": {}

},

"id": "3c4d5e6f-7g8h-9i0j-1k2l-3m4n5o6p7q8r",

"name": "Respond to Webhook",

"type": "n8n-nodes-base.respondToWebhook",

"typeVersion": 1,

"position": [

980,

300

]

},

{

"parameters": {

"text": "1. This webhook receives incoming requests with a 'question' field in the body.\n2. The webhook passes this question to the AI agent for processing.",

"height": 200,

"width": 400

},

"id": "4d5e6f7g-8h9i-0j1k-2l3m-4n5o6p7q8r9s",

"name": "Sticky Note",

"type": "n8n-nodes-base.stickyNote",

"typeVersion": 1,

"position": [

560,

160

]

},

{

"parameters": {

"text": "The AI Agent node processes the question using GPT-4o.\nIt's configured to provide helpful, accurate responses.",

"height": 200,

"width": 400

},

"id": "5e6f7g8h-9i0j-1k2l-3m4n-5o6p7q8r9s0t",

"name": "Sticky Note",

"type": "n8n-nodes-base.stickyNote",

"typeVersion": 1,

"position": [

760,

160

]

},

{

"parameters": {

"text": "This node returns the AI's response back to the original webhook caller.",

"height": 200,

"width": 400

},

"id": "6f7g8h9i-0j1k-2l3m-4n5o-6p7q8r9s0t1u",

"name": "Sticky Note",

"type": "n8n-nodes-base.stickyNote",

"typeVersion": 1,

"position": [

960,

160

]

}

],

"connections": {

"Webhook": {

"main": [

[

{

"node": "AI Agent",

"type": "main",

"index": 0

}

]

]

},

"AI Agent": {

"main": [

[

{

"node": "Respond to Webhook",

"type": "main",

"index": 0

}

]

]

}

},

"settings": {},

"staticData": null,

"pinData": {},

"versionId": "abcdef12-3456-7890-abcd-ef1234567890",

"triggerCount": 0,

"tags": []

}