# Your low code N8N guide for beginners

Theory Before Application

Written by: Hamadache Rania

# Preface

Months ago, I didn't really know what I was doing. For more than two years after graduating as a theoretical physicist, I worked whatever jobs I could find simply to earn money, without a clear vision of where I was going. Eventually, I quit my job. Being unemployed forced me to face an uncomfortable truth: if I stayed idle, my mind would slowly rot.

I tried filling the time with reading novels, something I usually enjoy, but even that failed to help. The sense of emptiness only grew. I felt stuck, drained, and directionless.
That was when I came across n8n.
At the time, I didn't know what it was, whether I would truly use it, or if I would succeed with it at all. But I knew one thing for certain: doing nothing would lead to regret. So I started anyway.
The beginning was not easy. I learned through trial and error, confusion, and persistence.
Slowly, things began to make sense. Over time, I regained confidence, not only in using n8n, but in my ability to learn, build, and solve problems again. I went on to teach n8n to others, help people with their projects, and build workflows for clients.
Because I enjoy learning through written guides, I decided to write this book for those who feel stuck, overwhelmed, or unsure where to begin, and for those who are just starting their journey with n8n. If you are reading this, chances are you are exactly where I once was.
I hope this guide helps you find clarity, confidence, and a place to start.

**Hamadache Rania**

# CONTENTS

# Chapter I:

# Definitions:

## 1) Overview:

In this chapter, we focus on introducing the basic definitions needed to understand the concepts discussed in this work. All definitions are presented in simple language so that readers with no prior experience in the field can easily follow along.

## 2) What Is Automation?

Automation is a low code process of using technology to perform tasks automatically with minimal human intervention.
So, if a person has to repeatedly perform the same tasks, such as copying data, sending emails, or updating systems, automation can handle them.

Automation exists to save time, reduce manual effort, and improve reliability, allowing people to focus on higher value work rather than repetitive operations.

## 3) Popular No-Code Automation Tools

| **Automation tools** | **Pros** | **Cons** |
|---|---|---|
| **Zapier: one of the most well-known no-code automation platforms** | Easy to use, beginner-friendly, huge library of integrations | Limited customization, paid plans can get expensive, not self-hosted. |
| **Make: a visual automation platform that lets you design workflows using a drag-and-drop interface** | Very visual, flexible, supports complex scenarios with branching logic | Can be overwhelming for beginners, cloud-based only (no self-hosting). |
| **n8n: (pronounced "n-eight-n") is open-source workflow automation tool, that uses a node-based system that is highly flexible, self-hosted, and extensible.** | Open-source, low-code/no-code, supports custom code, self-hosting for full control, integrates with APIs and AI tools | Slight learning curve compared to Zapier for absolute beginners, but much more powerful in the long run. |

## 4) Why Choose n8n?

(1) Open-Source and Self-Hosted You own your data and workflows. No dependency on a cloud service or subscription.

(2) Flexibility n8n allows you to integrate APIs, write small code snippets, and create complex workflows that platforms like Zapier cannot handle easily.

(3) Cost-Effective Since it's open-source, you can use it freely or host it yourself, avoiding high recurring costs.

(4) AI & Advanced Integration Ready n8n can connect to AI tools, databases, and external systems, making it future-proof for agentic workflows.

(5) Low-Code You don't need to be a developer to start; the visual interface allows you to build workflows quickly, while still giving power-users the ability to customize with code.

## 5) Workflow

A workflow is a structured sequence of steps that defines how tasks are executed from start to finish.

### 5).1 Traditional Workflow

A normal workflow follows a fixed and predictable path.

### 5).2 Agentic Workflow

An agentic workflow uses an AI agent capable of reasoning, deciding, and adapting its actions dynamically.

### 5).3 Traditional Workflow vs Agentic Workflow

| **Traditional Workflow** | **Agentic Workflow** |
|---|---|
| **Fixed steps** | dynamic steps |
| **Rule-based** | Goal driven |
| **Predictable** | Adaptive |
| **No reasoning** | AI reasoning |
| **Low risk** | Higher flexibility, higher complexity |

## 5).4 The Workflow components:

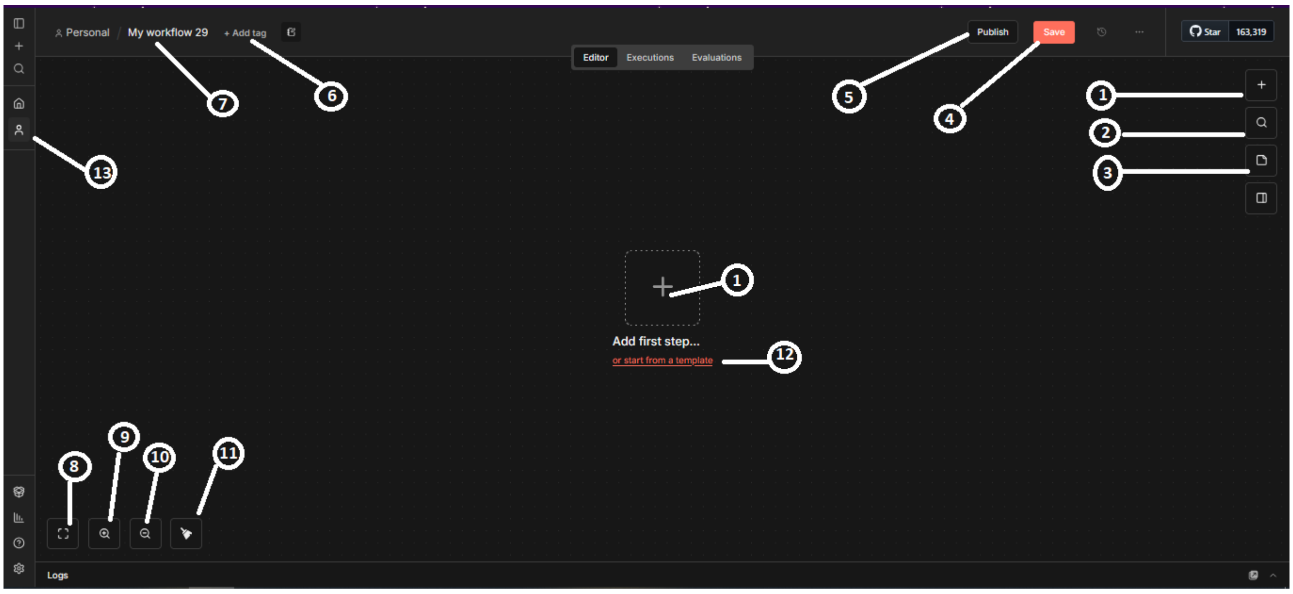


Figure I:.1: Workflow components

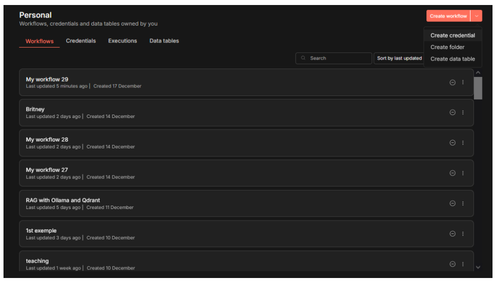
**Note: this is based on version 2.0.0**

1. Add a node
2. Search
3. Add a sticky note
4. Save the workflow
13.Personal

5. Publish the workflow (make it active)
6. Add a tag
7. Name of the workflow
8. Map of the workflow

9. Zoom in
10. Zoom out
11. Tidy up
12. Work using a template

## 5).5 Personal components:


Personal
Workflows, credentials and data tables owned by you

My workflow 29
Last updated 5 minutes ago | Created 17 December

Britney
Last updated 2 days ago | Created 14 December

My workflow 28
Last updated 2 days ago | Created 14 December

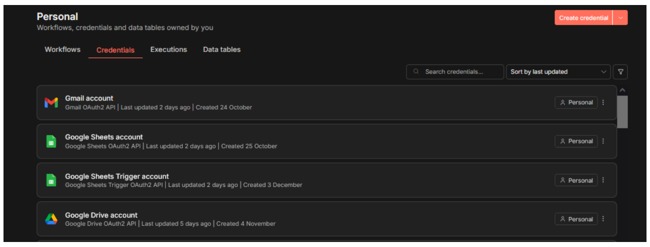
My workflow 27
Last updated 2 days ago | Created 14 December

RAG with Ollama and Qdrant
Last updated 5 days ago | Created 11 December

1st exemple
Last updated 3 days ago | Created 10 December

teaching
Last updated 1 week ago | Created 10 December

(a) Your saved workflows


Personal
Workflows, credentials and data tables owned by you

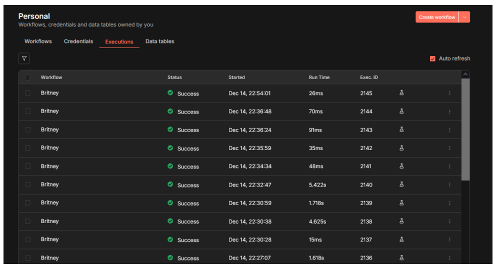
Gmail account
Gmail OAuth2 API | Last updated 2 days ago | Created 24 October — Personal

Google Sheets account
Google Sheets OAuth2 API | Last updated 2 days ago | Created 25 October — Personal

Google Sheets Trigger account
Google Sheets Trigger OAuth2 API | Last updated 2 days ago | Created 3 December — Personal

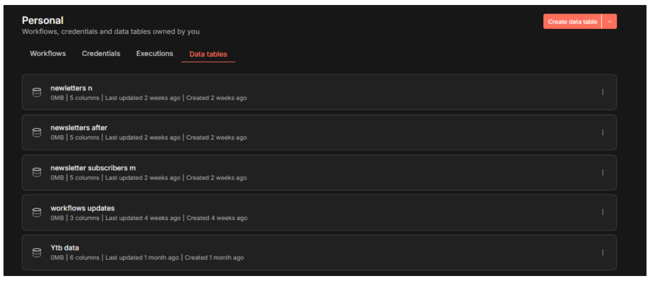
Google Drive account
Google Drive OAuth2 API | Last updated 5 days ago | Created 4 November — Personal

(b) Your saved credentials


Personal
Workflows, credentials and data tables owned by you

| Workflow | Status | Started | Run Time | Exec. ID |
|---|---|---|---|---|
| Britney | Success | Dec 14, 22:54:01 | 26ms | 2145 |
| Britney | Success | Dec 14, 22:36:48 | 70ms | 2144 |
| Britney | Success | Dec 14, 22:36:24 | 91ms | 2143 |
| Britney | Success | Dec 14, 22:35:59 | 35ms | 2142 |
| Britney | Success | Dec 14, 22:34:34 | 48ms | 2141 |
| Britney | Success | Dec 14, 22:32:47 | 5.422s | 2140 |
| Britney | Success | Dec 14, 22:30:59 | 1.718s | 2139 |
| Britney | Success | Dec 14, 22:30:38 | 4.625s | 2138 |
| Britney | Success | Dec 14, 22:30:28 | 15ms | 2137 |
| Britney | Success | Dec 14, 22:27:07 | 1.618s | 2136 |

(c) Your workflow executions


Personal
Workflows, credentials and data tables owned by you

newletters n
0MB | 5 columns | Last updated 2 weeks ago | Created 2 weeks ago

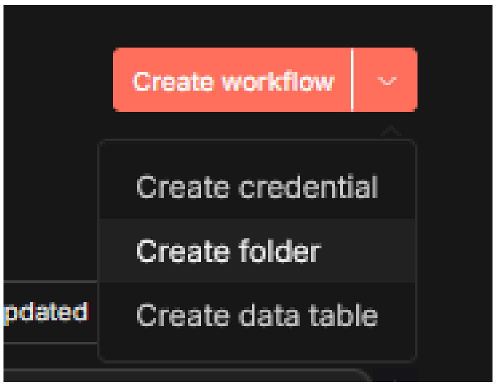
newsletters after
0MB | 5 columns | Last updated 2 weeks ago | Created 2 weeks ago

newsletter subscribers m
0MB | 5 columns | Last updated 2 weeks ago | Created 2 weeks ago

workflows updates
0MB | 3 columns | Last updated 4 weeks ago | Created 4 weeks ago

Ytb data
0MB | 6 columns | Last updated 1 month ago | Created 1 month ago

(d) Your saved data tables



(e) Create a workflow/a folder/ a data table

Figure I:.2: Overview of the personal

## 5).6 Data Table:

A data table is a structured way of organizing, storing, viewing, and managing data information into rows and columns, where each row represents a single record and each column represents a specific data field, it's like an n8n sheet.

## 5).7 Credentials:

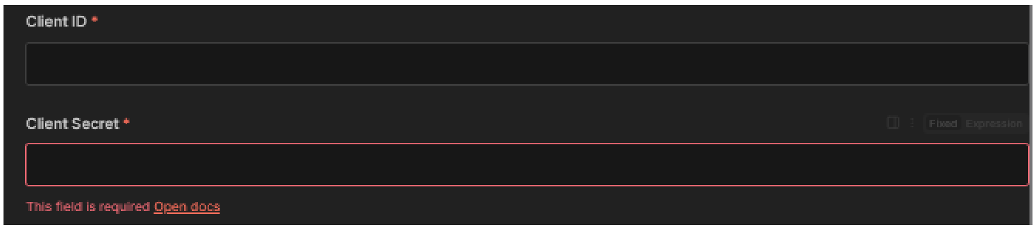
Credentials are secure pieces of information used to authenticate and authorize access to external services, applications, or APIs. They typically include items such as API keys, access tokens, usernames, and passwords.

**Credential Types:**

**Client ID and Client Secret:**
These credentials are used for nodes that require you to create an application on an external platform.

Examples include Google nodes (Gmail, Google Sheets, etc.) and Microsoft nodes.

They allow secure authentication using OAuth and let n8n act on your behalf.

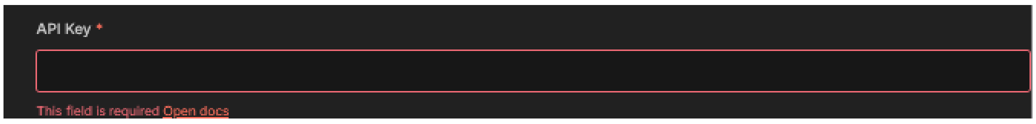

Client ID *

Client Secret *

This field is required Open docs

Figure I:.3: Client ID and Client Secret

**API Key:**
An API key is a unique identifier provided by a service to authenticate requests. It is usually simpler than OAuth and is commonly used for services that do not require user level permissions.


API Key *

This field is required Open docs

Figure I:.4: API Key

**Access Token:**
An access token is a temporary credential generated after successful authentication (often via OAuth). It grants limited-time access to an API and is used by n8n to perform authorized actions securely.

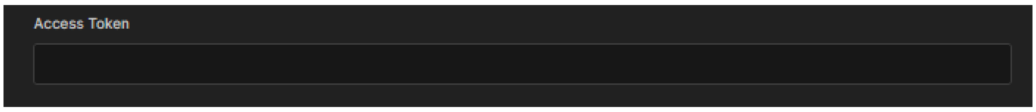

Access Token

Figure I:.5: Access Token

## 6) URL:

Uniform Resource Locator, is the address used to locate and access a resource on the internet. **The URL components:**

| Part | Meaning |
|---|---|
| Protocol | How the request is sent |
| Domain | Where the server is located |
| Port | Optional; the server port |
| Path | The specific resource or endpoint |
| Query parameters | Extra data sent to filter or customize the request |
| Fragment / anchor | Optional, points to a section inside the resource (mostly for web pages) |

## 7) Local Host:

Refers to your own computer/server.

http://localhost:5678

Figure I:.6: n8n Local Host

I:.6 represents the n8n local host usually using the port 5678.

## 8) HTTP & HTTPS

### 8).1 HTTP:

HyperText Transfer Protocol

### 8).2 HTTPS:

HyperText Transfer Protocol Secure

### 8).3 Difference between HTTP and HTTPS:

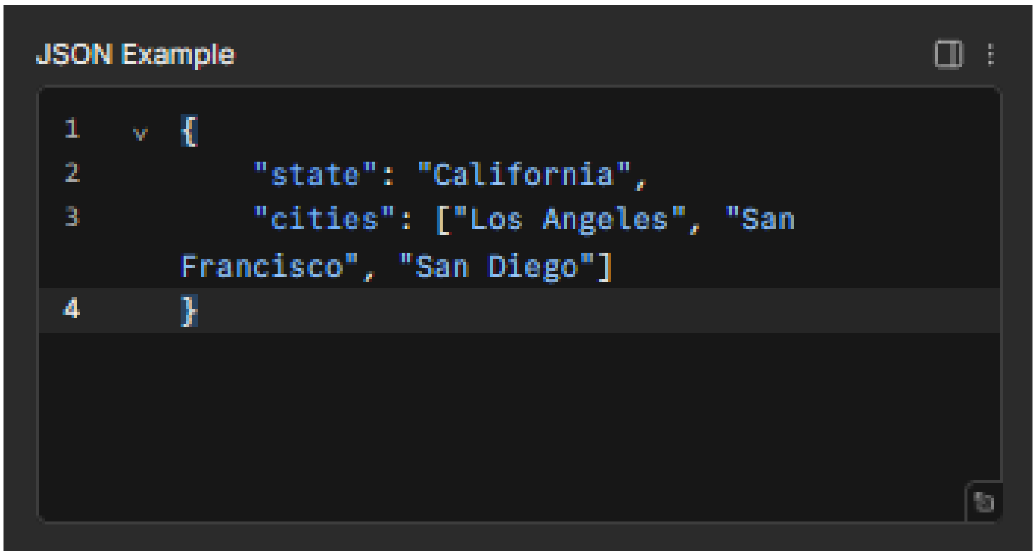
| Feature | HTTP | HTTPS |
|---|---|---|
| Security | None (Plain text) | Encrypted (TLS/SSL) |
| Data protection | No | Yes |
| Use case | Internal (Local Host) | Production and sensitive data |

## 9) JSON:

JSON (JavaScript Object Notation) is a standard data format used to store and exchange information between systems.
It is designed to be easy for humans to read and easy for machines to process, which is why it is widely used in

web applications, APIs, and modern software systems.

- Every JSON object must start with { and end with }, these are called **keys**.
- Every array must start with [ and end with ], these are called **index**.
- If a value is a string, it must be enclosed in double quotes " ".
- If there are multiple values, separate them with a comma,.
- If there is only one value, do not add a comma after it.



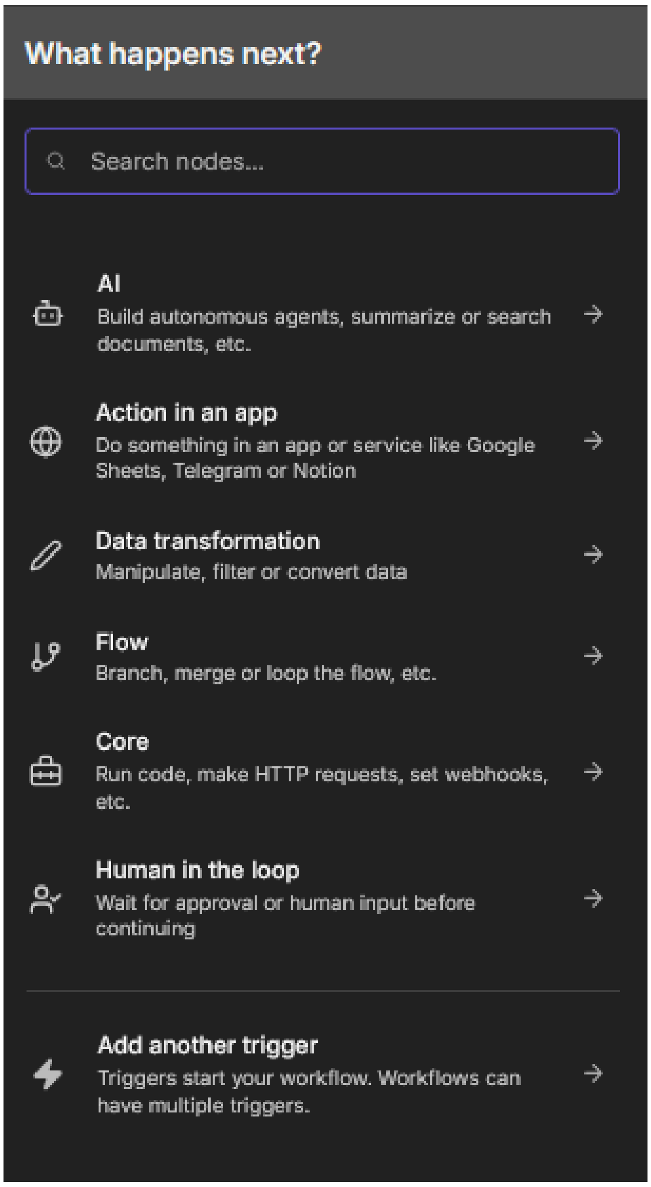
```
{
    "state": "California",
    "cities": ["Los Angeles", "San
Francisco", "San Diego"]
}
```

Figure I:.7: A JSON example

## 10) Nodes:

A node is a single unit or step in an n8n workflow : it represents an action, trigger, or service.

In a workflow, adding a new node is done by clicking the plus (+) button, and every workflow must begin with a trigger, which defines how and when the automation starts.



What happens next?

Search nodes...

**AI**
Build autonomous agents, summarize or search documents, etc.

**Action in an app**
Do something in an app or service like Google Sheets, Telegram or Notion

**Data transformation**
Manipulate, filter or convert data

**Flow**
Branch, merge or loop the flow, etc.

**Core**
Run code, make HTTP requests, set webhooks, etc.

**Human in the loop**
Wait for approval or human input before continuing

**Add another trigger**
Triggers start your workflow. Workflows can have multiple triggers.

Figure I:.8: Different type of nodes

## 10).1 Artificial Intelligence:

Refers to systems that can analyze data, classify information, generate text, summarize content, recognize patterns, and make decisions or generate outputs in a way that mimics human reasoning.

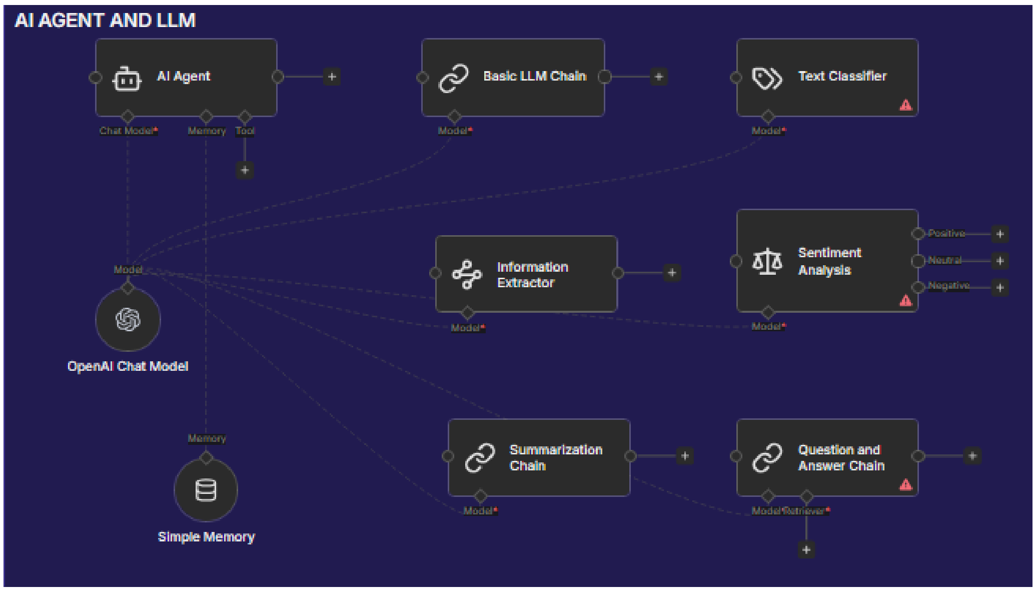


Figure I:.9: Different AI nodes

**Large Language Model (LLM)**

A Large Language Model (LLM) is an AI model trained on vast amounts of text data to understand, generate, and manipulate human language. LLMs can answer questions, write text, summarize information, translate languages, and analyze content, but they do not act on their own.

An LLM is **the Brain**, it has no built in awareness of goals, tools, or workflows.

**AI Agent**

An AI agent is a system that uses one or more AI models (often an LLM) to pursue a goal autonomously. An agent can plan actions, make decisions, use tools (APIs, databases, workflows), and adapt its behavior based on outcomes.

The AI agent is **the decision maker** .

All the AI nodes need a **Chat model**, whether is an AI agent or a LLM.

## 10).2 Parser in the AI:

In an AI agent, a parser plays a critical role in connecting human language, AI output, and automated actions.
An AI agent often receives unstructured input and also produces unstructured output.
The parser is responsible for turning this unstructured content into structured data that the agent can act on.
Adding the parser or not is your choice depends on the data received and sent.

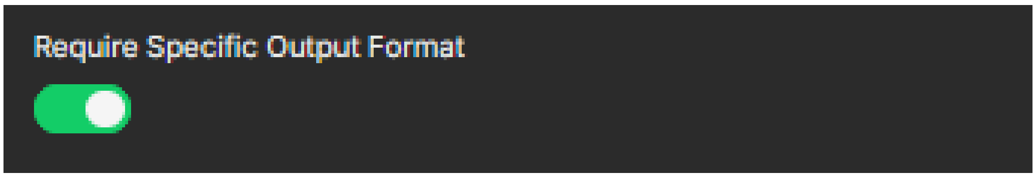

Require Specific Output Format

Figure I:.10: Turning on the parser

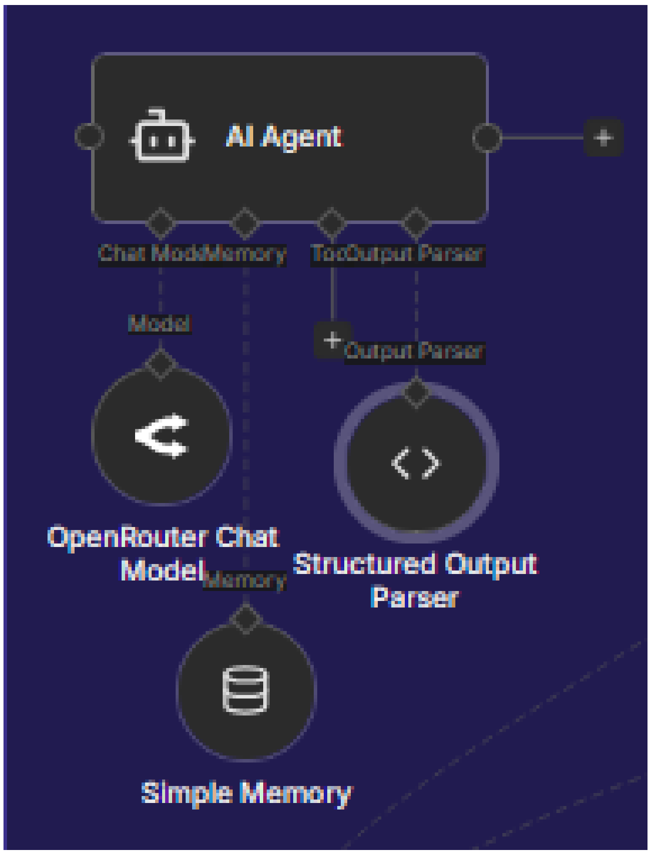


Figure I:.11: The parser node in the AI agent

## 10).3 Action in app:

An action is a specific operation performed within an application as part of a workflow. Examples include sending an email, creating a database record, updating a spreadsheet, or posting a message to a chat platform.

## 10).4 Data transformation:

is the process of modifying data as it moves through a workflow. This can include cleaning, formatting, filtering, enriching, or converting data into a form that can be used by the next step in the automation.

## 10).5 Flow:

A flow is the logical path that data follows through a workflow, from the trigger to the final action. It defines the order of execution and how information moves between nodes.

## 10).6 Core:

The core refers to the central engine that manages node execution, data handling, error control, and communication between different services.

## 10).7 Human in the loop:

Human in the loop describes where human input or approval is required at certain stages of an automated process.

## 10).8 Triggers:

A trigger is the event that starts a workflow.

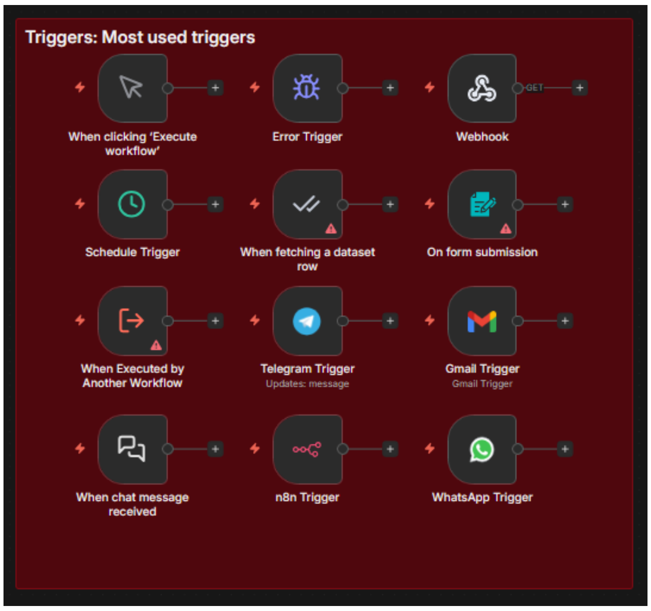

Triggers: Most used triggers
When clicking 'Execute workflow'
Error Trigger
Webhook
GET
Schedule Trigger
When fetching a dataset row
On form submission
When Executed by Another Workflow
Telegram Trigger
Updates: message
Gmail Trigger
Gmail Trigger
When chat message received
n8n Trigger
WhatsApp Trigger

Figure I:.12: Most used Triggers

## 10).9 Most used triggers in n8n:

To start any workflow in n8n, you must use a trigger.

Figure I:.12 shows the most commonly used triggers in n8n workflows. There are many triggers available, but in this guide we will focus on the ones you are most likely to use.

Below are some of the most important triggers:

**Manual Trigger:** This trigger allows you to run a workflow manually with a single click. It is mainly used for testing and debugging workflows, as it requires no configuration.

**Schedule Trigger:** This trigger runs a workflow automatically at a specific time or interval that you choose. For example, you can schedule a workflow to run every day at 9:00 AM.

**Execute Workflow Trigger:** This trigger allows one workflow to call and execute another workflow. It is useful when you want to split a complex automation into smaller, reusable workflows.

**Chat Trigger:** This trigger allows your workflow to start when a message is sent in a chat interface.

**Telegram and WhatsApp Triggers:** These triggers execute the workflow when a message is received from a user. In this guide, we focus only on message-received events, although these platforms support many other trigger types.

**Gmail Trigger:** Similar to messaging triggers, this trigger starts the workflow when an email is received in a Gmail account.

**n8n Trigger (Fetching Workflows):** This trigger is used to monitor and fetch information about workflows inside n8n itself. We will mainly use it for fetching and monitoring workflows.

**Data Table Trigger (Fetching Rows):** This trigger retrieves data from rows inside a selected data table. It is useful when your workflow needs to process stored data automatically.

**Error Trigger:** This trigger runs when an error occurs in a workflow. It is commonly used for automatic error handling.

**On Form Submission Trigger:** This trigger allows you to create a form with a title and input fields, send it to users, and start the workflow once the form is submitted.

**Webhook Trigger:** A webhook allows n8n to receive or send data from external systems, especially websites or services that do not have a dedicated n8n node. Any trigger that receives data from outside n8n such as forms, Telegram, or WhatsApp—is conceptually based on webhook communication.

## 10).10 Data Handling Nodes:

**Set Node:** it is used to create, edit, or remove data fields without writing any code.
It allows you to define values manually, rename fields, format data, or prepare data before sending it to another node. This node is ideal for no-code users who want full control over their data structure( you can also use Json in it)

**Code Node:**The Code node allows you to manipulate data using JavaScript Or Python.
It is used when more advanced logic is required, such as complex conditions, loops, calculations, or custom transformations that cannot be easily handled with no-code nodes.

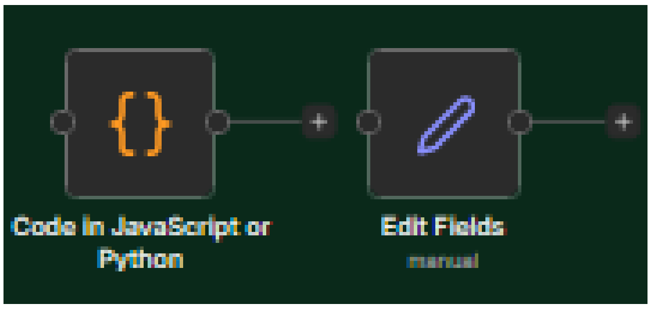


Figure I:.13: Set and Code node

## 10).11 Filtering and Data Control Nodes:

**Filter Node:** allows you to control which data continues in the workflow by applying conditions. You can define rules such as “only continue if this value exists,” “only if a number is greater than X,” or “only if a status equals success.”

**Remove Duplicates Node:** it is used to eliminate repeated data items based on one or more fields. It is especially useful when working with lists, databases, or API responses where duplicate entries may exist in order to have leaner data and prevents repeated actions.

**Limit Node:** restricts the number of items that continue through the workflow. For example, you can limit execution to the first 5 or 10 items instead of processing an entire dataset.

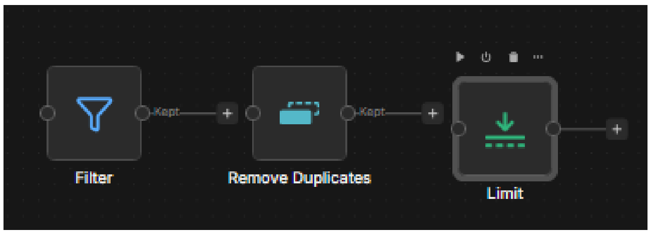


Figure I:.14: Set and Code node

## 10).12 Logic and Flow Control Nodes:

**If Node:**

The If node is used to create conditional logic in a workflow.
for example: value = 5,

If condition: value is greater or equal to 5, and then sends the data to one of two branches:

True (condition is met, which is the case in this example)

False (condition is not met)

**Data Types:**

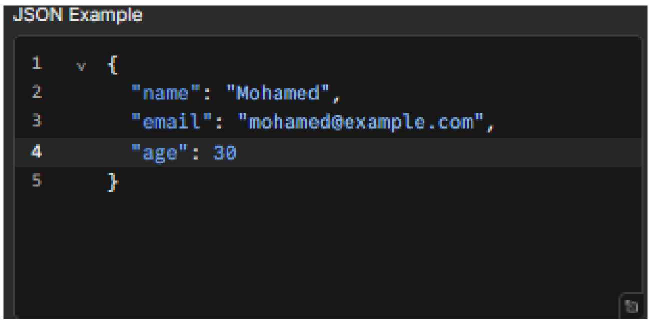
**1. String:** plain text, ex: ”Hello, world!”

**2. Boolean:** is a data type that can only have two values: true or false.

**3.Array:** is an ordered list of values stored together, ex: [”apple”, ”banana”, ”cherry”], [1, 2, 3, 4, 5].

**4.Binary:** is used for Files and raw data like :Images, PDFs, audio, attachments.

**5.Object:** is a structured data that contains key-value pairs. ex in json:


JSON Example

```
{
    "name": "Mohamed",
    "email": "mohamed@example.com",
    "age": 30
}
```

Figure I:.15: Object in JSON example

**Switch Node:** it is an advanced alternative to the If node. Instead of only two paths, it allows you to route data into multiple branches based on different values or conditions.
For example, depending on a status field (pending, approved, rejected), the workflow can follow a different path for each case, and also create n path for the n option ( here n=3 so 3 paths for 3 options)

**Merge Node:** it is used to combine data from multiple branches back into a single flow.
It is commonly used after If or Switch nodes when different paths need to rejoin later in the workflow.
The Merge node can combine data in different ways, such as merging items, appending data, or choosing data from one branch.

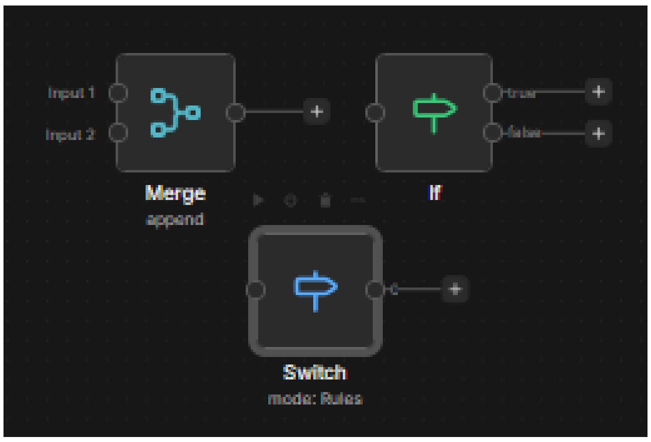


Figure I:.16: Logic and Flow Control Nodes

## 10).13 Data Processing Nodes:

**Aggregate Node:** is used to combine multiple data items into a single summary.
Common operations include: counting items and grouping data by a specific field.

**SplitInBatches / Split Node:** it divides data into smaller parts or batches.
Use it when: You need to process large datasets step by step.
Each batch is processed sequentially or in parallel, depending on your workflow configuration.

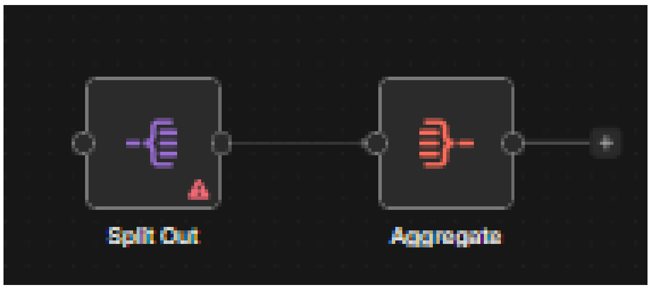


Figure I:.17: Data Processing Nodes

## 10).14 File Handling Nodes:

**1. Extracting Data from Files(csv, PDF, RTF, text file, XML, XLS, XLSX, JSON, ODS, ICS, HTML ):**

1. Import the file into the workflow.

2. Then use the node to parse it into structured JSON objects for further processing.

3. After reading, you can process, analyze, or send them to other systems.

**2. Compressing and Decompressing Files:**

**a. Compress Node:**

Combine multiple files into a single ZIP or GZIP archive.

**b. Decompress Node:**

Extract files from a ZIP or GZIP archive to process each file individually in the workflow.

**3. Converting Data to Files (csv, RTF, text file, XML, XLS, XLSX, JSON, ODS, ICS, HTML ):**

You can simply convert your data into any fill supported in the node.

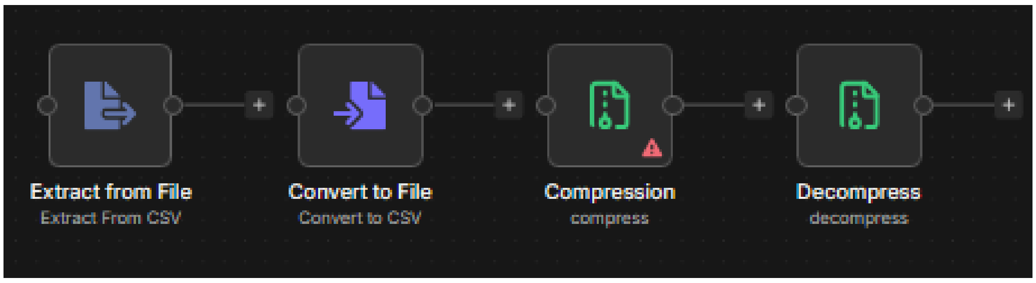


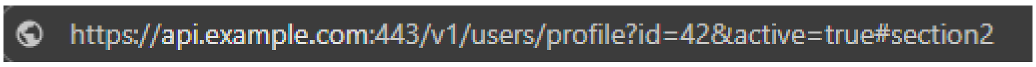
Figure I:.18: File Handling Nodes

# Chapter II:

# Webhhok and Http request

## 11) Decoding a URL:

Read URL definition in section(6))



https://api.example.com:443/v1/users/profile?id=42&active=true#section2

Figure II:.1: URL example

- Protocol: is represented by the **https://** in other cases **http://**
- Domain: is **api.example.com**
- Port: **:443**
- Path: **/v1/users/profile**
- Query parameters: **?id=42&active=true**
- Fragment / anchor: **#section2**

## 12) Webhook and respond to webhook node:

**In this subsection, we focus primarily on the Webhook node and provide a clear and comprehensive explanation of its role, emphasizing its importance within the workflow**

Read the Webhook definition in section (10).9).

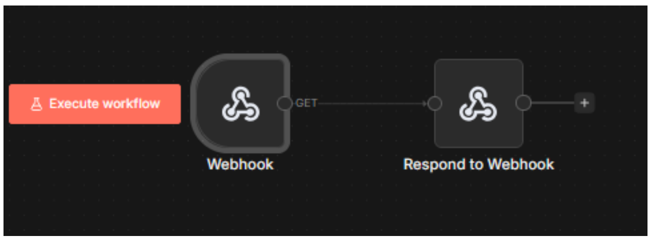


Figure II:.2: Webhook and respond to webhook

## 12).1 Webhook URLs

When calling a website, we need a URL to send the request, whether we are retrieving data or sending data.

For this purpose, we usually work with two types of URLs:
Test URL and Production URL

These URLs follow the same structure, The main difference between them is that the test URL includes a test indicator, example:
For testing: **https://example.com/webhook/(Path)**
For production: **https://example.com/webhook-test/(Path)**
During development and testing, we always use the test URL.
Once everything is working as expected, we switch to the production URL, which is used in real-world scenarios and handles actual data.

## 12).2 Webhook and Local Host:

A webhook is a server to server HTTP request.
When an external service tries to send data to your webhook, it requires a public URL. If the webhook is hosted locally, the request will fail because your machine is not exposed to the internet. Such an external service is referred to as a third party.

## 12).3 HTTP methods in webhook:

There are many HTTP methods that can be used with webhooks, in this subsection, we will focus on the most frequently used methods, which are:

(1) **GET** Retrieve data from a server.

(2) **POST** Send data to a server.

(3) **DELETE** Remove data from a server.

(4) **PUT / PATCH** Update existing data.

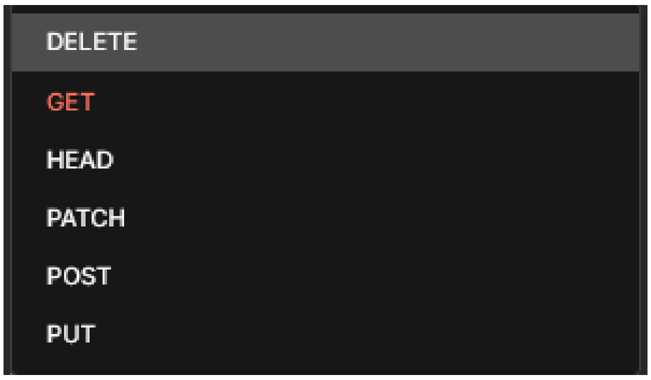


Figure II:.3: HTTP methods

### 12).4 Allow multiples HTTP methods:

While most websites indicate which HTTP method to use, some do not. To handle this, we can enable “Allow Multiple HTTP Methods” in the node settings, allowing the webhook to receive both GET and POST requests.

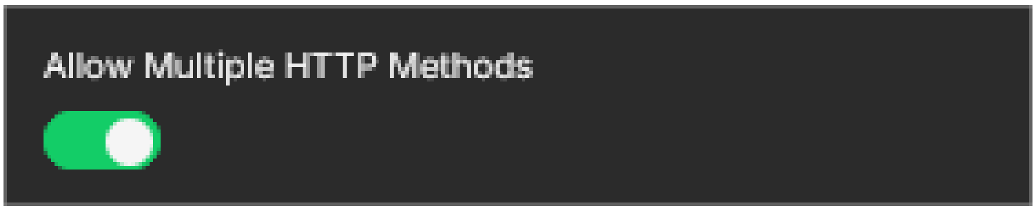

Allow Multiple HTTP Methods

Figure II:.4: Allow Multiple HTTP Methods

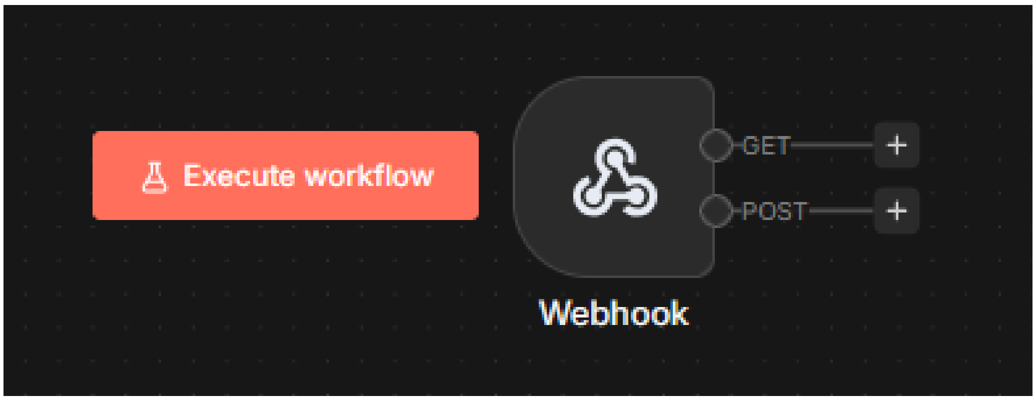
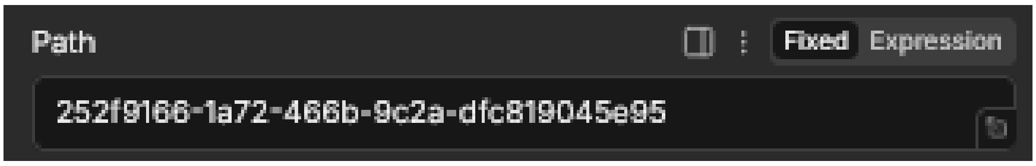


Figure II:.5: webhook get and post

### 12).5 Path:

The webhook URL includes a path, as discussed in Section ( 12).1) and section ( 6)). This path can be easily customized (which is recommended) or left unchanged.


Path

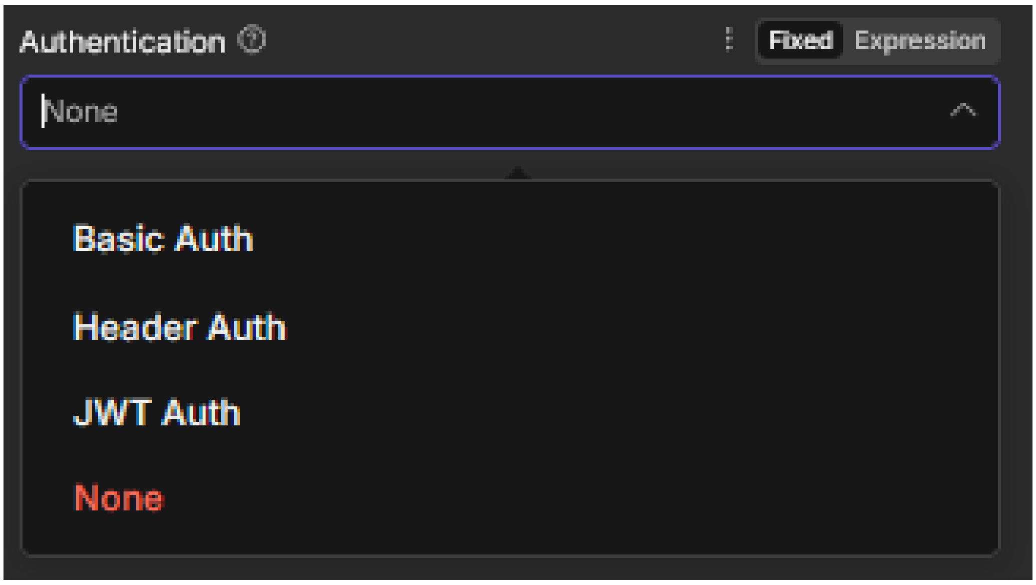
Fixed Expression

252f9166-1a72-466b-9c2a-dfc819045e95

Figure II:.6: Webhook Path

### 12).6 Authentication:

To secure your webhook, you can enable an authentication method (this step is optional). One of the most commonly used methods is Basic Authentication, which requires a username and password that only the receiver should know, preventing unauthorized access to the webhook.


Authentication

Fixed Expression

None

Basic Auth

Header Auth

JWT Auth

None

Figure II:.7: Authentication

### 12).7 Respond to Webhook:

The Respond to Webhook node sends a response back to the service that triggered the webhook.

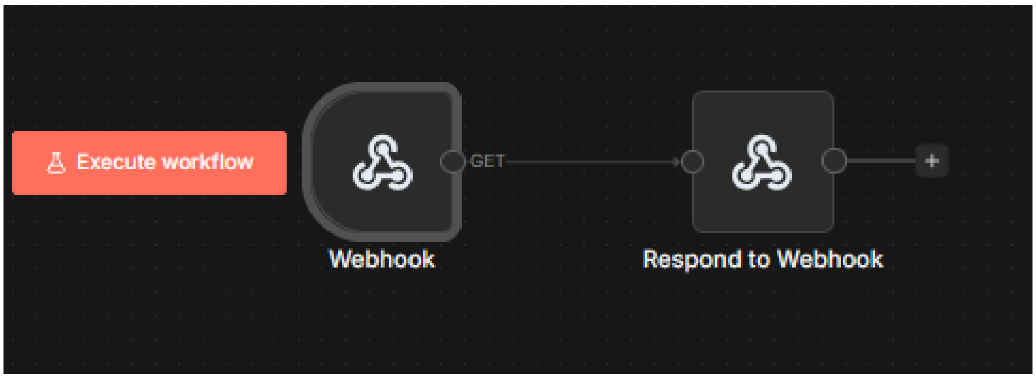


Figure II:.8: Respond to Webhook node

## 13) The HTTP Request node:

The HTTP Request node is used to send requests to external services or APIs and receive responses from them.

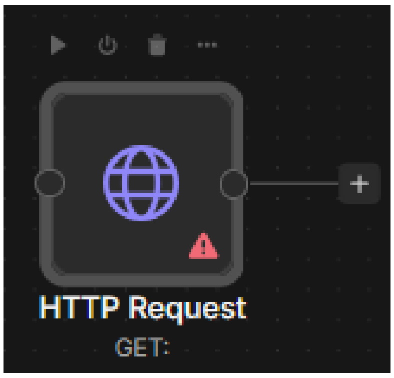


Figure II:.9: HTTP request node

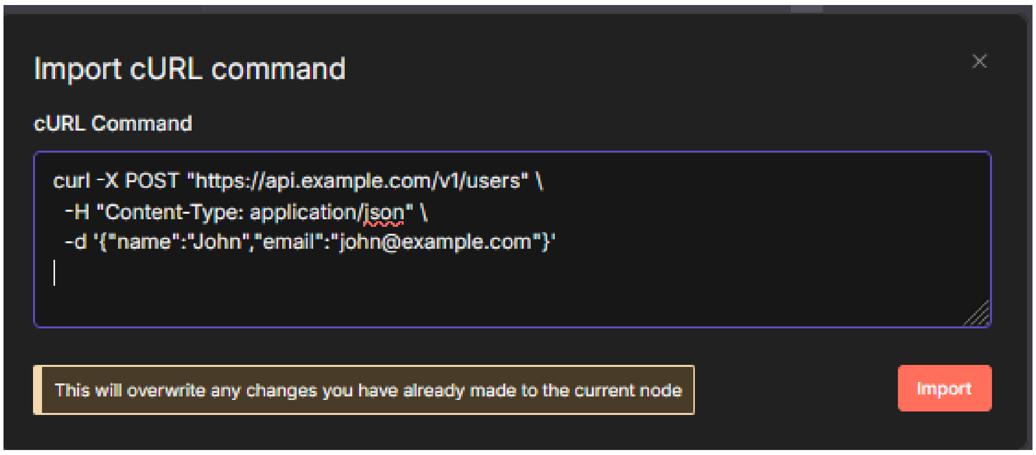
### 13).1 cURL:

cURL (client URL) is a command line tool used to send HTTP requests(Call APIs, Send data to servers...) to a URL and receive responses from a server.

In many cases, an API or service provides a cURL command as part of its documentation.

A cURL command already contains:The HTTP method, The URL, Headers (such as authentication),Query parameters, Request body (if any)

By pasting the cURL command into the import cURL, the configuration is automatically filled.


Import cURL command
cURL Command
This will overwrite any changes you have already made to the current node
Import

```
curl -X POST "https://api.example.com/v1/users" \
  -H "Content-Type: application/json" \
  -d '{"name":"John","email":"john@example.com"}'
```

Figure II:.10: cURL example

### 13).2 Decoding the cURL:

The figure (II:.10) presents a cURL example, which can be simplified as follows:

- **-X**: Specifies the HTTP method (in this case, `POST`).
- **URL**: https://api.example.com/v1/users
- **-H**: Defines request headers (`Content-Type`, `Authorization`).
- **-d**: Sends the request body (data).
- **Query parameters**: Appended directly to the URL which is not used in this example (check section (6)) and section(12).1)).

### 13).3 Manual configuration:

If a cURL command is not provided, the request must be configured manually. In this case, all the details that would normally be included in the cURL command (see Section 13).2) must be specified directly in the node, strictly following the API documentation.

## 13).4 Difference between a Webhook and HTTP request:

**Webhook VS HTTP Request:**

| Webhook | HTTP Request |
|---|---|
| Trigger node | Action node |
| Event Driven | Manual request sent to a server |
| Requires a publicly accessible URL to receive data(webhooks does not work on localhost, does listen). | Can be sent to any accessible URL (HTTP Requests works on localhost or public URLs, does not require a listening server) |

# Chapter III:

# JSON in N8N:

In n8n, almost all communication between nodes happens through **JSON**.
Every node receives input as JSON and outputs JSON.

APIs, webhooks, AI models, databases, all of them use JSON.

When you add a **Code node**, you are reading, transforming, and returning JSON manually using JavaScript.

That's why understanding JSON is not optional in n8n. It's a survival skill.

Read the JSON definition and basics in (9)), and also JSON example in (I:.7)

Note: **This section is designed for beginners with little to no coding background. The goal is not to teach programming, but to help you understand the structure of JSON, learn how to read and interpret errors, and confidently fix common issues when pasting JavaScript or code generated by AI tools.**

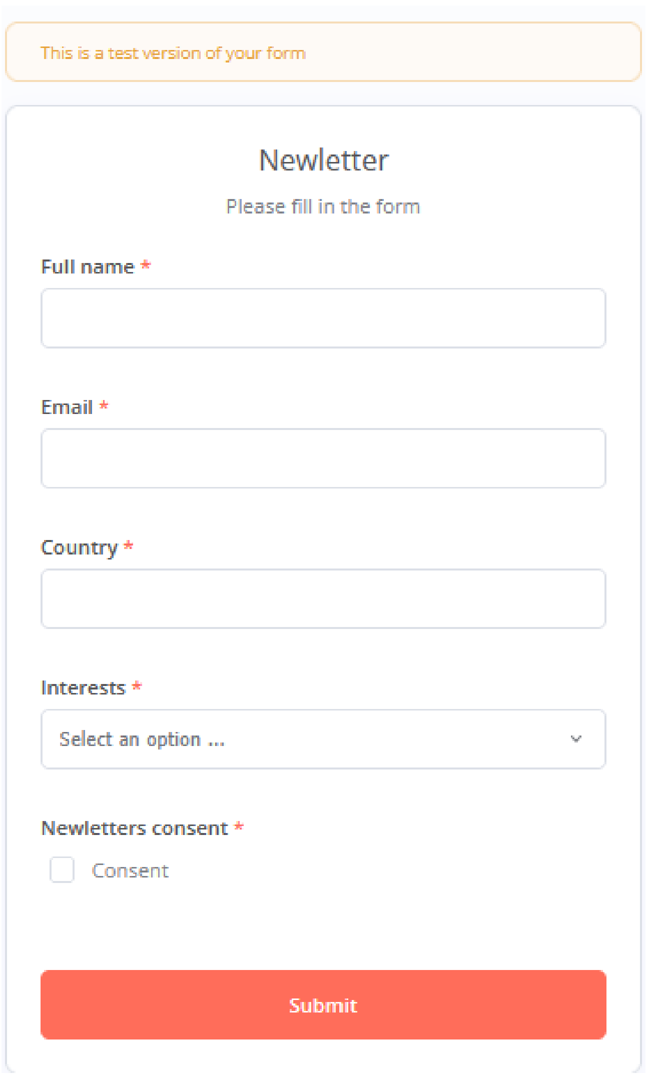
## 14) Understanding JSON by a workflow:

Let's build a small, practical workflow.
A user fills out a form, and the submitted data is received as JSON.
Our goal is to append this data to a spreadsheet, but before doing so, we need to structure and clean the data.

To achieve this, we will use a Code node with JavaScript.
The JavaScript code itself will be generated by an AI tool, and we will focus on understanding, validating, and fixing that code rather than writing it from scratch.

### 14).1 Form node:

The idea is to create a newsletter subscription form.
The workflow starts with a Form Trigger, where the user fills in the required information.
This form submission acts as the trigger for the workflow and produces the initial JSON input that will be processed by the following nodes.


This is a test version of your form

Newletter

Please fill in the form

Full name *

Email *

Country *

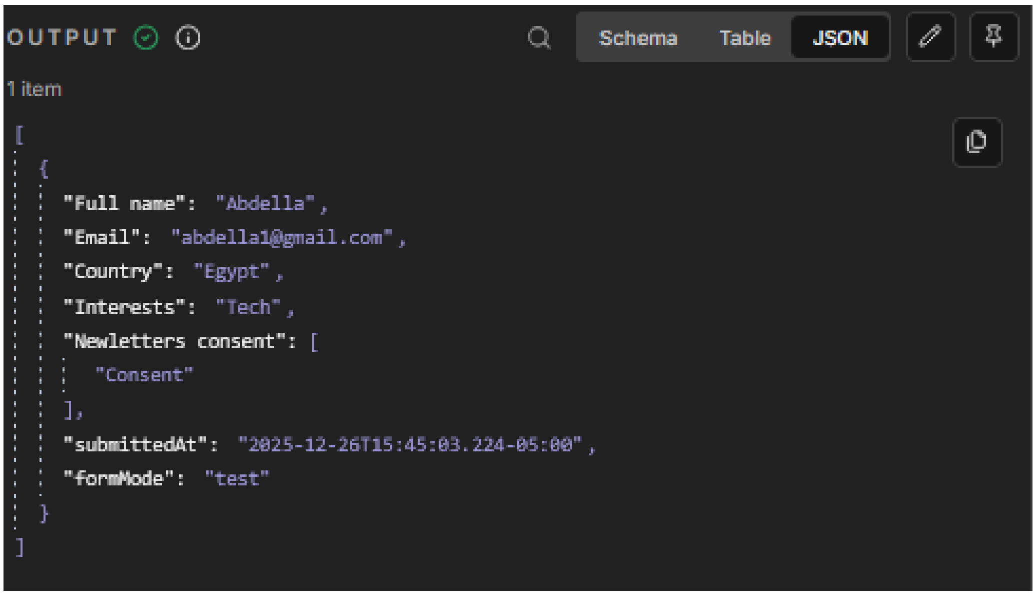
Interests *

Select an option ...

Newletters consent *

Consent

Figure III:.1: Form to be submitted



```
[
  {
    "Full name": "Abdella",
    "Email": "abdella1@gmail.com",
    "Country": "Egypt",
    "Interests": "Tech",
    "Newletters consent": [
      "Consent"
    ],
    "submittedAt": "2025-12-26T15:45:03.224-05:00",
    "formMode": "test"
  }
]
```

Figure III:.2: The form output in JSON

**Decoding the JSON Format: a nested array–object JSON structure**

Figure (III:.2) shows the JSON output produced by the form. Using Section (9)), we will decode this structure step by step:

- The output starts with an index, which means the data is wrapped inside an **array**.
- Inside the array, each item is represented as a **JSON object** composed of key–value pairs.
- The `Newsletters consent` field itself contains an **array** as its value.
- **This means the output is an array that contains an object, and that object contains another array.**
- We call this structure **a nested array–object JSON structure**

**A nested JSON structure** is a JSON format in which arrays and objects are placed inside one another.

**Note: Sometimes JSON is just an object, and sometimes it is just an array. In more complex cases, both appear together. When arrays and objects are placed inside one another, the JSON structure is called nested.**

## 14).2 Accessing Values from JSON in a Code Node:

In the Code node, the form output is first stored in a variable (commonly named **data**).

**To access any value, you must always start from this variable.**

If we want to access "Egypt" from the Country field, we must start with data.
Since data is an array, we need to include an index. Each index corresponds to an object inside the array.

In this example, there is only one object, so the index is 0 (remember, arrays in JavaScript always start at 0).

We can then access the value like this:

**data[0].Country**

Now lets do the same for Abdellah,

1. Start with data, The root of your JSON is an array, so we need an index:**data[0]**

2.Look at the key: The value "Abdella" is under the key:"Full name"

**Notice: If the key has a space, dot notation will NOT work, We must use bracket notation ( the bracket notation can also be used if the key has no space, its safe to use it in both cases).**

3. Access the value: **data[0]["Full name"]**

This returns: "Abdella".

**Start with data, pick the array index [0], access the key using brackets if it has spaces (dot notation works only if no spaces).**

If the input starts with an object instead of an array, there is no need for an index.

Let's take the example in Figure (I:.15). Here, the JSON structure is a simple object, and the keys are written normally, with no spaces.

To access the name "Mohamed", you can simply write: **data.name**

The same applies to email and age.

**Important: The keys must be written exactly as they appear in the JSON structure, including capitalization.**
If I want to access **"Consent"** under the "Newletters consent" key, applying what we have learned so far, I would write:

**data[0]["Newletters consent"][0]**

**Quick explanation:** The data starts with an array that contains only one object, which is why the index is 0. The key "Newletters consent" contains a space, so it must be written in bracket notation. Finally, this key contains an array with one item, which is why we also use [0] to access its value.

## 14).3 Code node:

**Role of the Code Node:**

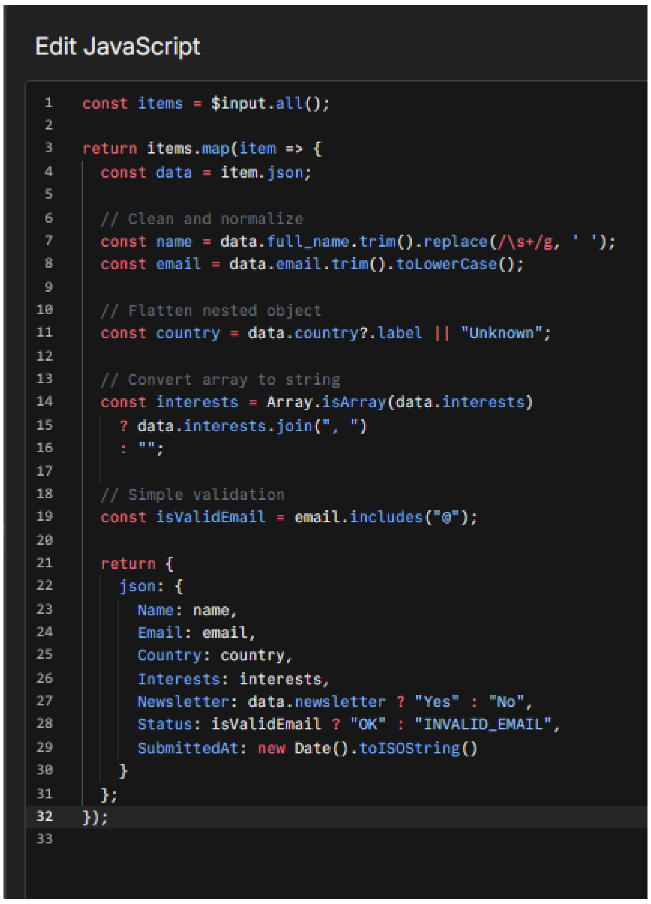
The purpose of the Code node is to process and prepare the form data before it is sent to Google Sheets. Specifically, we use it to:

- Validate the submitted data
- Normalize the data (clean names and standardize formats)
- Convert selected interests into a readable string
- Append clean and structured data to Google Sheets
- Flag invalid submissions without breaking the workflow

All of these steps are performed using AI-generated JavaScript inside a Code node, with the focus on understanding and verifying the JSON structure rather than writing code from scratch.

## 14).4 Reading the code node:

I explained my goal to the AI tool and asked it to write a code for me. I also provided information about my form and the input structure. Despite that, this is the code it generated:



```
const items = $input.all();

return items.map(item => {
  const data = item.json;

  // Clean and normalize
  const name = data.full_name.trim().replace(/\s+/g, ' ');
  const email = data.email.trim().toLowerCase();

  // Flatten nested object
  const country = data.country?.label || "Unknown";

  // Convert array to string
  const interests = Array.isArray(data.interests)
    ? data.interests.join(", ")
    : "";

  // Simple validation
  const isValidEmail = email.includes("@");

  return {
    json: {
      Name: name,
      Email: email,
      Country: country,
      Interests: interests,
      Newsletter: data.newsletter ? "Yes" : "No",
      Status: isValidEmail ? "OK" : "INVALID_EMAIL",
      SubmittedAt: new Date().toISOString()
    }
  };
});
```

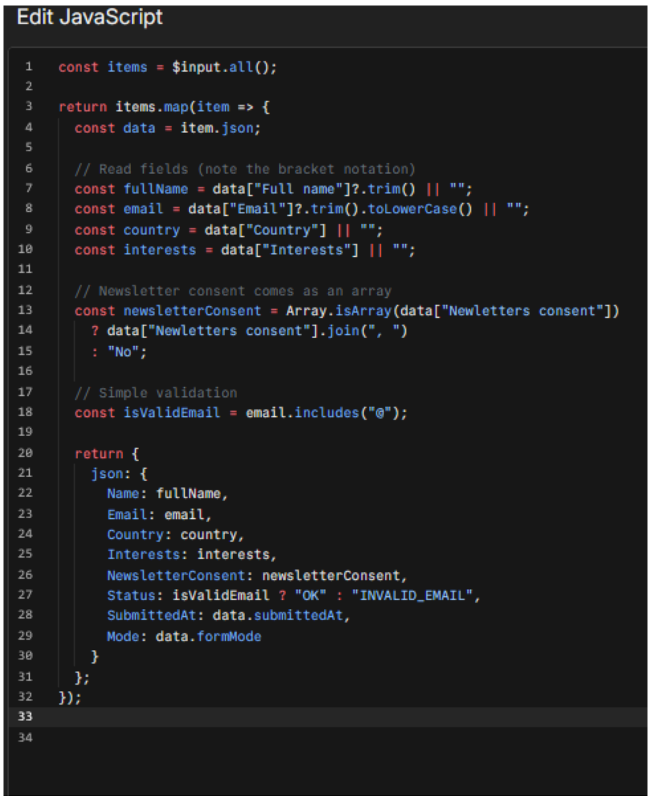
Figure III:.3: Code generated by AI

Before hitting Execute, we can see that the data has been treated as an object rather than an array (there is no index next to data, as we explained earlier). This only happens when the JSON starts with an object.

We can also notice that the field names are not written exactly as they appear in the form (which they should be). In addition, ”Full name” and ”Newletters consent” are not written correctly and are not treated as keys with spaces—they are not enclosed in brackets as required for such keys.

In this case, the code must be fixed immediately. The easiest way to do this is to copy the Code node input in JSON and send it to the AI tool. The AI can then generate a corrected version of the code based on the actual structure of the data.

**This way, you can spot errors before executing and also determine whether the code you are using is correct or needs to be fixed.**



Edit JavaScript

```javascript
const items = $input.all();

return items.map(item => {
  const data = item.json;

  // Read fields (note the bracket notation)
  const fullName = data["Full name"]?.trim() || "";
  const email = data["Email"]?.trim().toLowerCase() || "";
  const country = data["Country"] || "";
  const interests = data["Interests"] || "";

  // Newsletter consent comes as an array
  const newsletterConsent = Array.isArray(data["Newletters consent"])
    ? data["Newletters consent"].join(", ")
    : "No";

  // Simple validation
  const isValidEmail = email.includes("@");

  return {
    json: {
      Name: fullName,
      Email: email,
      Country: country,
      Interests: interests,
      NewsletterConsent: newsletterConsent,
      Status: isValidEmail ? "OK" : "INVALID_EMAIL",
      SubmittedAt: data.submittedAt,
      Mode: data.formMode
    }
  };
});


```

Figure III:.4: Code after fixation

Now, Figure (III:.4) shows that the AI has updated some of the details based on the JSON that was sent to it. We can see that data is now treated as an array, and the keys "Full name", "Newletters consent", and all other keys are written using bracket notation (see 14).2; brackets can be used in both cases).

Now, when we hit Execute, the code will run, and we will obtain the output.

# Chapter IV:

# Prompt engineering:

## 15) Definitions:

**Prompt Engineering** is the practice of designing, structuring, and refining inputs (prompts) given to large language models and AI systems in order to obtain accurate, relevant, and reliable outputs. It focuses on guiding the model's behavior through clear instructions, context, constraints, and examples, while minimizing errors such as hallucinations and inconsistent responses.

To ask an AI agent to perform a task, we must provide an input known as a prompt. The prompt is written by a human and serves as the instruction or request. The AI agent then processes this prompt using its underlying model, commonly a large language model (LLM), and generates a response, referred to as the output.

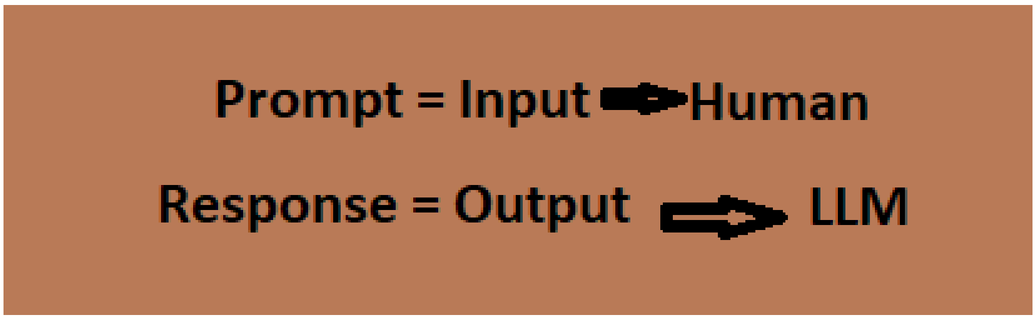


Figure IV:.1: Input and Output

## 16) Schema of a prompt:

A prompt can be as simple as sending a direct task to a LLM and expecting an output, and in many cases, the model will respond.
However, to obtain accurate, consistent, and reliable results, the prompt must be structured with some extra detail.

Providing additional elements helps the model better understand the task and reduces ambiguity. A well-designed prompt typically includes:

**Context:** Background information that frames the task.

**Task (Action):** A clear and explicit instruction describing what the model must do.

**Format:** The expected structure of the output.

**Examples:** Sample inputs and outputs that guide the model's behavior.

The most important element in this schema is the task, as it defines the action the model must perform.

While examples and output format are optional in some cases, they become essential in others especially when precision, consistency, or specific structure is required.

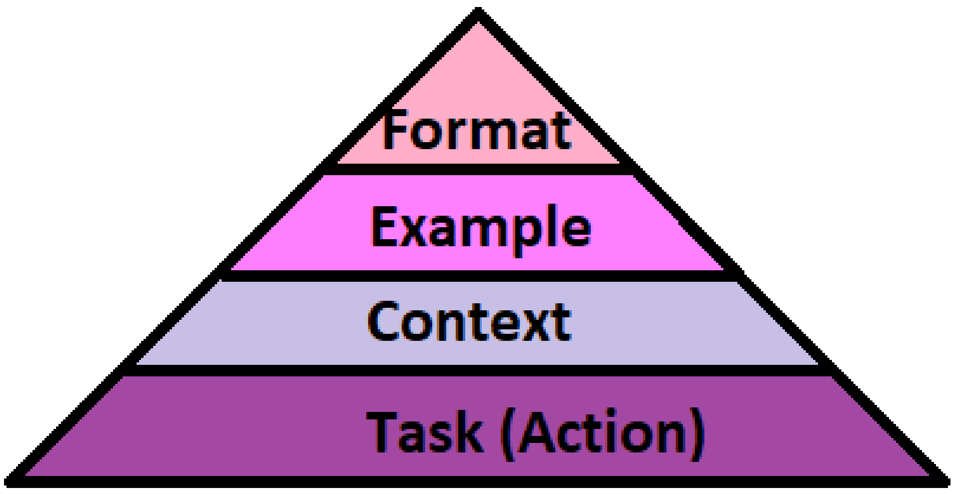


Figure IV:.2: Prompt diagram

## 16).1 Text Classification:

Text classification is the process of assigning one or more labels to a piece of text based on its content. It is one of the most common and practical applications of LLMs.

Although large language models are not specifically designed for text classification, they are still highly capable of performing this task. Because LLMs learn general language patterns, context, and semantics, they can infer categories and labels from text.

This flexibility allows LLMs to handle text classification effectively through prompt-based instructions, despite not being purpose-built classifiers.

**How LLMs handle text classification:**

- Reads the input text

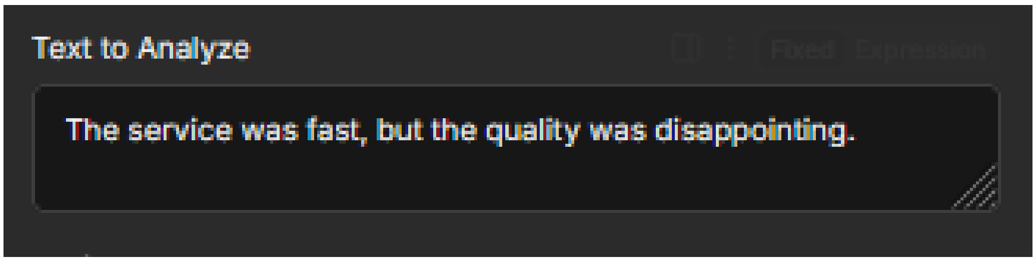
- Understands its context and intent
- Matches it to predefined categories
- Outputs the most relevant label

This can be done without training a new model, simply by using well-designed prompts.

**Common Text Classification Tasks:**

**Sentiment analysis (positive, negative, neutral):**

Using the LLM sentiment analysis.


Text to Analyze

The service was fast, but the quality was disappointing.

Figure IV:.3: Sentiment Analysis input

**When using the Sentiment analysis node there's no need to add a prompt.**

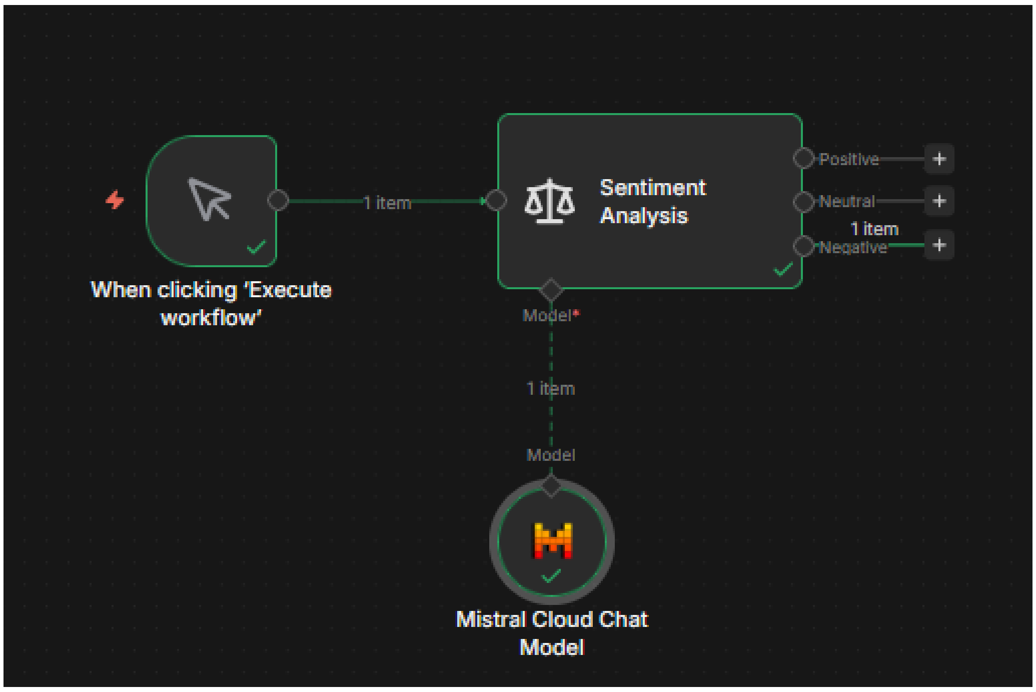
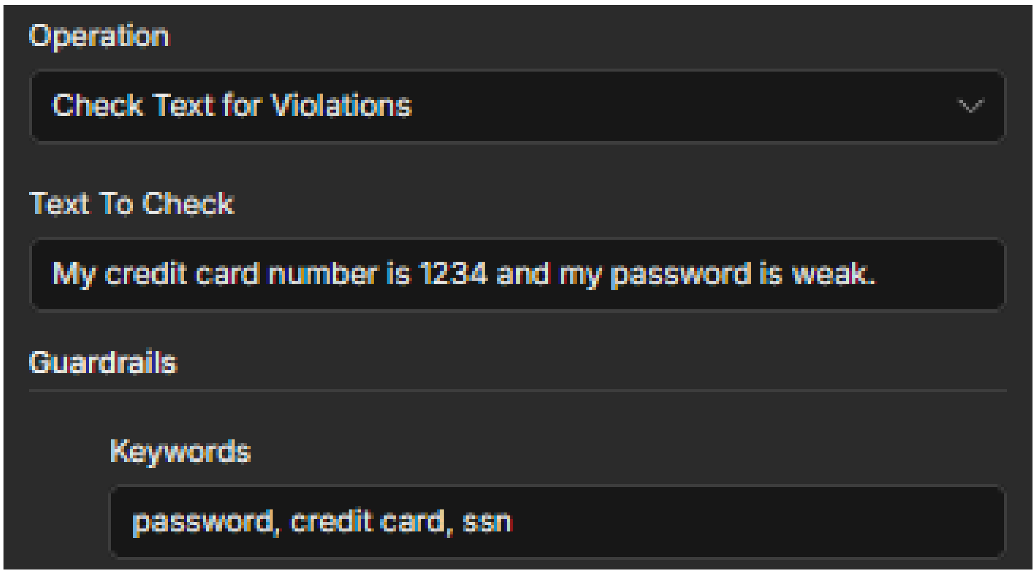


Figure IV:.4: Sentiment Analysis workflow

**Figure (IV:.4) shows the output of the input (IV:.3) in green which is negative**

**Check text for violation:**

using Guardrails node no need for a prompt


Operation

Check Text for Violations

Text To Check

My credit card number is 1234 and my password is weak.

Guardrails

Keywords

password, credit card, ssn

Figure IV:.5: Check text for violation input

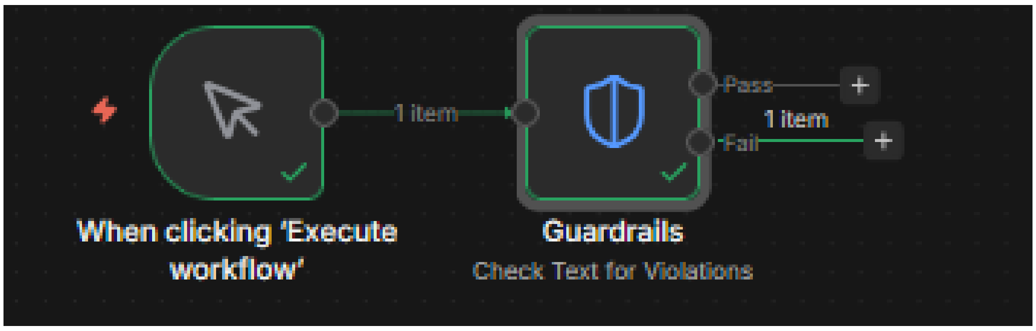
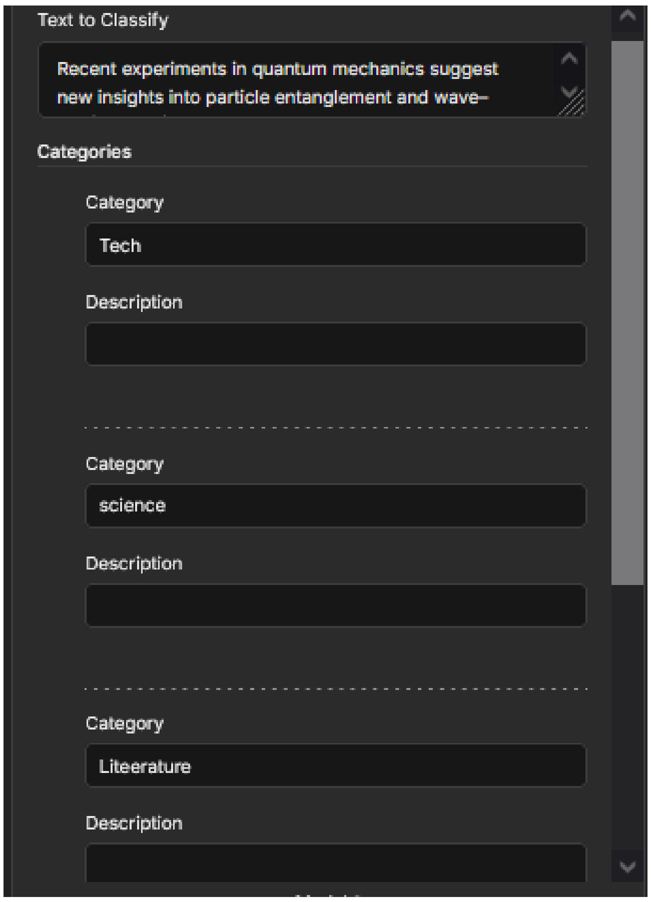


Figure IV:.6: Guardrails workflow

**Figure (IV:.6) shows that the input (IV:.5) fails validation because it matches one or more of the specified keywords.**

**Text classification:**

Using the Text classifier node.


Text to Classify

Recent experiments in quantum mechanics suggest new insights into particle entanglement and wave-

Categories

Category
Tech

Description

Category
science

Description

Category
Liteerature

Description

Figure IV:.7: The text classifier input

**In this node, we must provide the text to be classified along with the classification options. A brief description can also be added for each option to improve clarity and classification accuracy.**

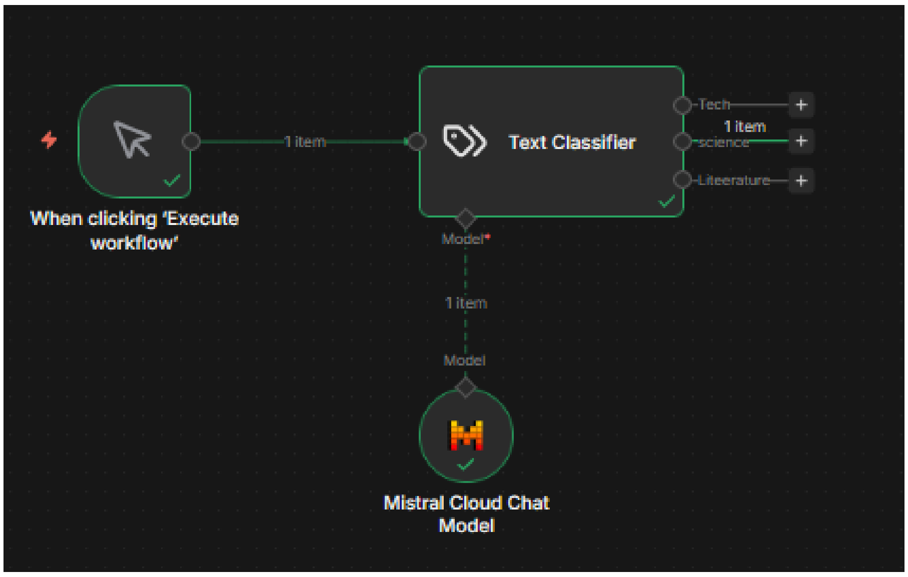


Figure IV:.8: Text classifier workflow

**Using the input (IV:.7), the figure (IV:.8) shows that the text was classified as a science text**

# 17) Prompting and Model Customization:

## 17).1 Pre-trained Models:

A pre-trained model is a model that has already been trained on large and diverse datasets to learn general language patterns, grammar, and knowledge.
Instead of training a model from scratch, developers reuse these models and adapt them to specific tasks such as classification, summarization, or question answering.

## 17).2 Fine-Tuning:

Fine-tuning is **the process** of further **training** a **pre-trained model** on a smaller, task specific dataset. This allows the model to specialize and consistently perform a particular task that requires high precision.

## 17).3 Zero-Shot Prompting:

Zero-shot prompting refers to asking a model to perform a task without providing any examples (Zero). The model relies solely on its pre-trained knowledge and the instructions given in the prompt.

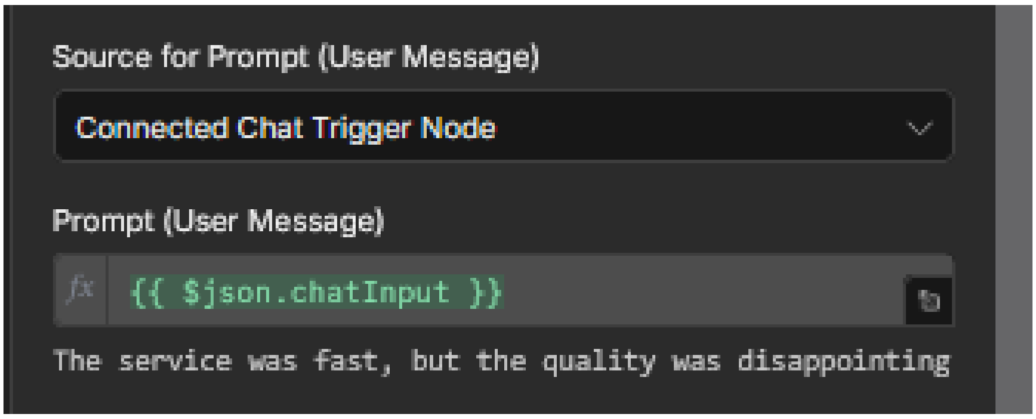
Let's take the input (IV:.3) and convert it into an LLM. We need to create a prompt specifically designed for sentiment analysis.

**Dynamic input in the AI agent and LLM:**

**This subsubsection can be applied to other workflows that follow the same pattern.**

When working with dynamic input from a previous node in an AI agent or LLM, first determine whether the previous node is a chat trigger. If it is, no additional input selection is needed. If it is another type of node, select Define Below and drag and drop the input you want the AI to read from the schema on your left. Next, go to Add Option and select System Message, this is where you write your prompt.

When writing your prompt, you may want to include the text you wish to analyze. To do this, drag and drop the text from the schema into your system message, where it will be displayed in green.



Source for Prompt (User Message)

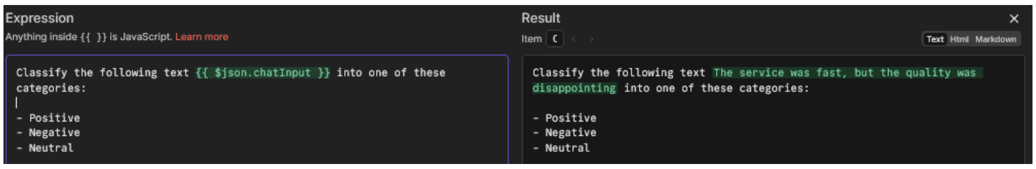
Connected Chat Trigger Node

Prompt (User Message)

```
{{ $json.chatInput }}
```

```
The service was fast, but the quality was disappointing
```

Figure IV:.9: Connect your AI to chat trigger

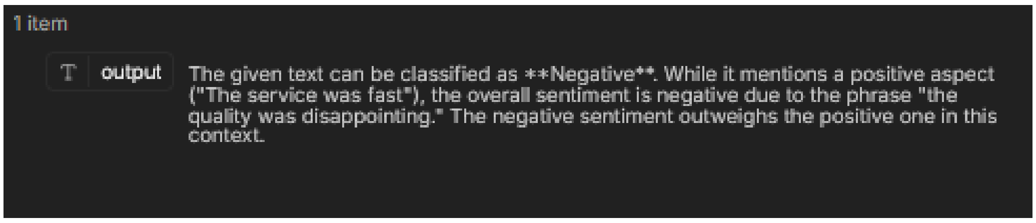

Expression
Anything inside {{ }} is JavaScript. Learn more

```
Classify the following text {{ $json.chatInput }} into one of these
categories:

- Positive
- Negative
- Neutral
```

Result
Item

```
Classify the following text The service was fast, but the quality was
disappointing into one of these categories:

- Positive
- Negative
- Neutral
```

Figure IV:.10: Connect your AI to chat trigger


1 item

output

```
The given text can be classified as **Negative**. While it mentions a positive aspect
("The service was fast"), the overall sentiment is negative due to the phrase "the
quality was disappointing." The negative sentiment outweighs the positive one in this
context.
```

Figure IV:.11: The output

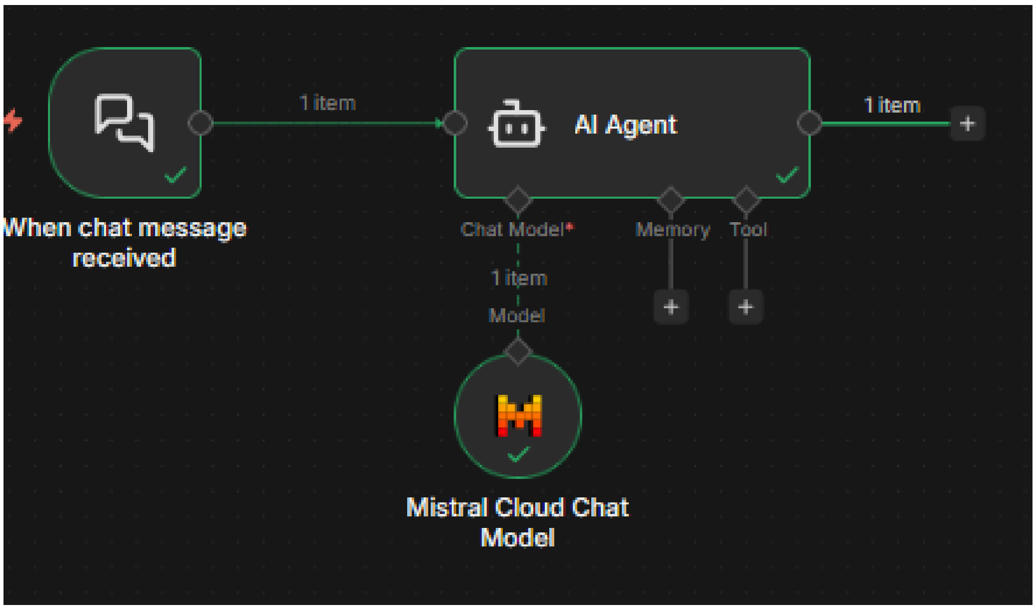
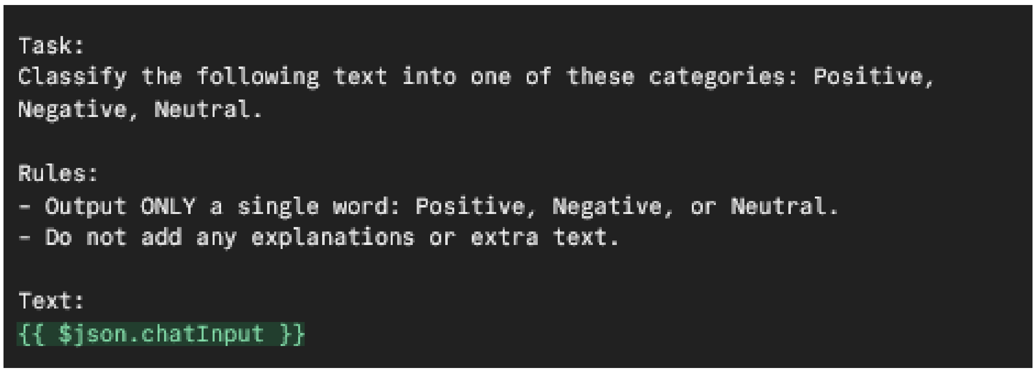


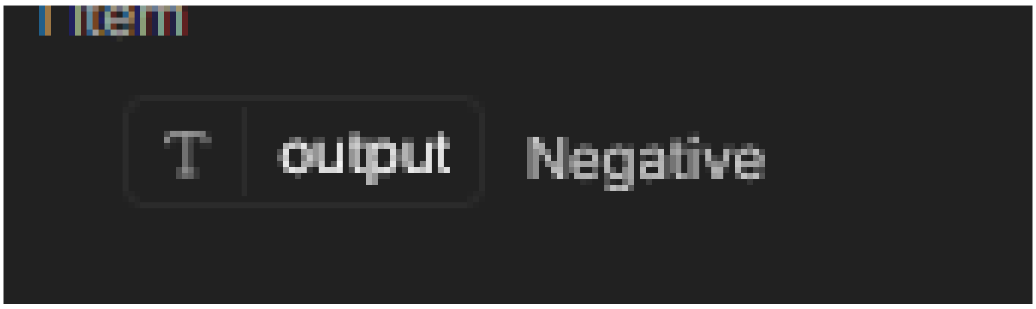
Figure IV:.12: The Workflow

**Figure (IV:.11) shows the result as "Negative," but it is written within a paragraph. To address this, we can refine the prompt and instruct the AI to respond with a single word.**



```
Task:
Classify the following text into one of these categories: Positive,
Negative, Neutral.

Rules:
- Output ONLY a single word: Positive, Negative, or Neutral.
- Do not add any explanations or extra text.

Text:
{{ $json.chatInput }}
```

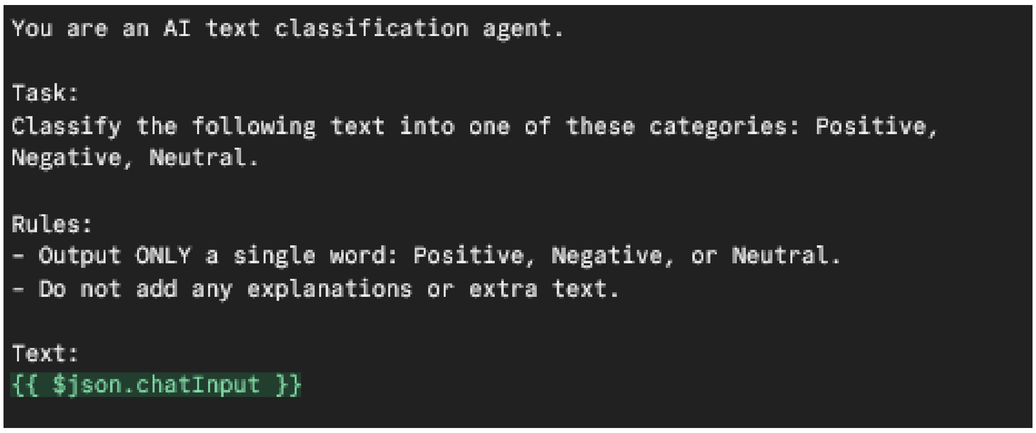
Figure IV:.13: The refined prompt



```
Negative
```

Figure IV:.14: The one word output

**For greater precision and accuracy, it is advisable to assign the AI a specific role , such as a teacher, a writer, or an assistant.**



```
You are an AI text classification agent.

Task:
Classify the following text into one of these categories: Positive,
Negative, Neutral.

Rules:
- Output ONLY a single word: Positive, Negative, or Neutral.
- Do not add any explanations or extra text.

Text:
{{ $json.chatInput }}
```

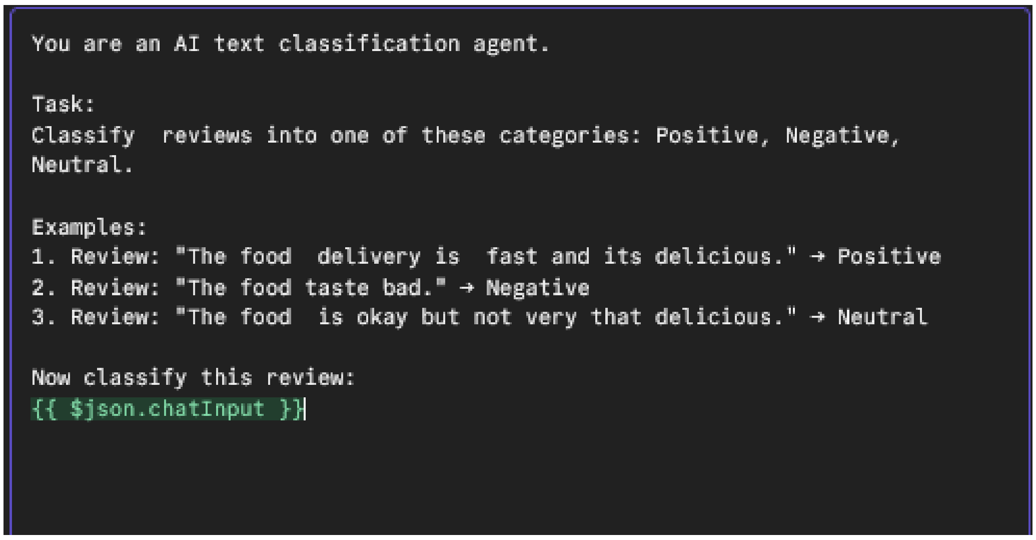
Figure IV:.15: The prompt with the role

**In conclusion, adding more details to your prompt and assigning a specific role to the AI can strengthen it and improve the accuracy of the output.**

This prompt, as constructed so far, is considered a **zero-shot prompt** because it does not include any examples.

## 17).4 Few-Shot Prompting:

Few-shot prompting improves performance by providing the model with at least one example that demonstrate the desired behavior to guide the model toward more accurate outputs.



```
You are an AI text classification agent.

Task:
Classify  reviews into one of these categories: Positive, Negative,
Neutral.

Examples:
1. Review: "The food  delivery is  fast and its delicious." → Positive
2. Review: "The food taste bad." → Negative
3. Review: "The food  is okay but not very that delicious." → Neutral

Now classify this review:
{{ $json.chatInput }}
```

Figure IV:.16: Few-Shot Prompt

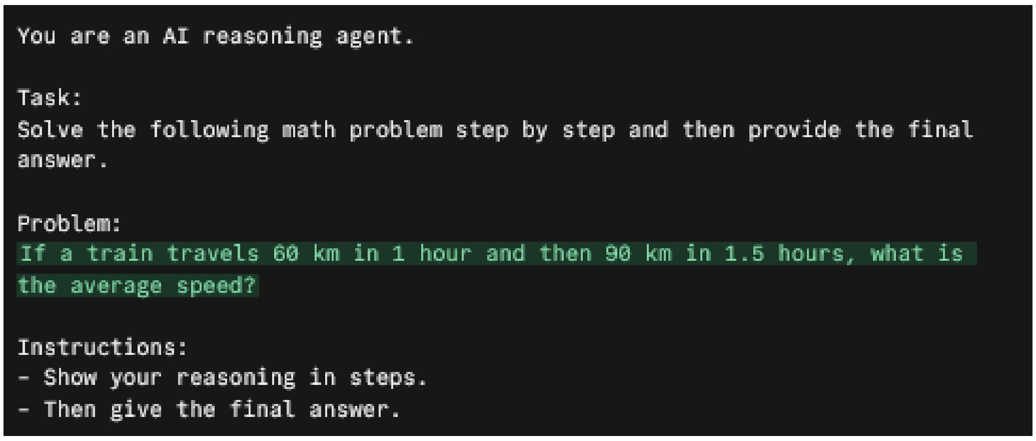
**By taking the prompt (IV:.15) and adding more examples, this prompt becomes a Few-Shot Prompt.**

## 18) Chain of Thought (CoT):

Chain of Thought (CoT) is a prompting technique that encourages a language model to explain its reasoning **step by step** before providing the final answer. This approach improves accuracy on tasks that require .

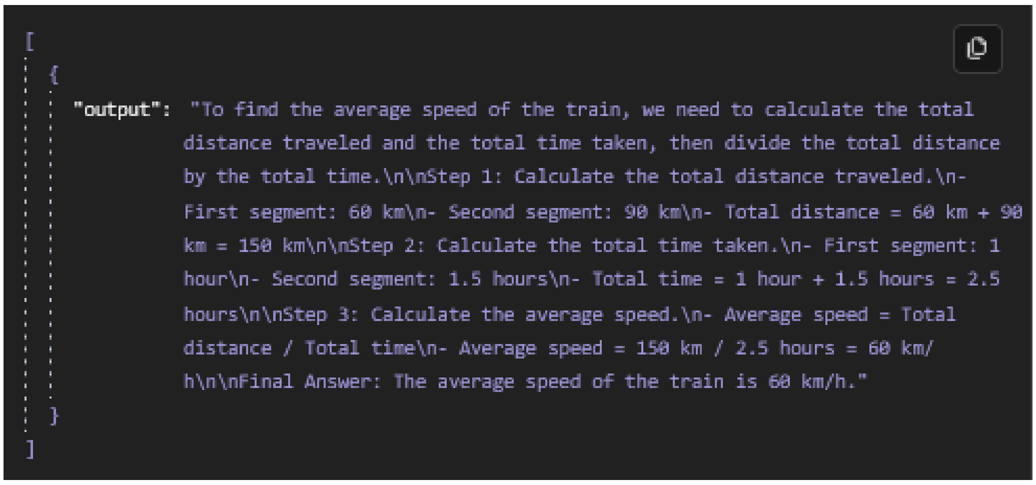
### 18).1 Zero-Shot CoT:

Using the zero-shot CoT method helps the LLM **think slowly** and break the task into steps for better reasoning. However, **the steps will be generated according to the model's own logic and choice.**



```
You are an AI reasoning agent.

Task:
Solve the following math problem step by step and then provide the final
answer.

Problem:
If a train travels 60 km in 1 hour and then 90 km in 1.5 hours, what is
the average speed?

Instructions:
- Show your reasoning in steps.
- Then give the final answer.
```

Figure IV:.17: Zero-Shot CoT

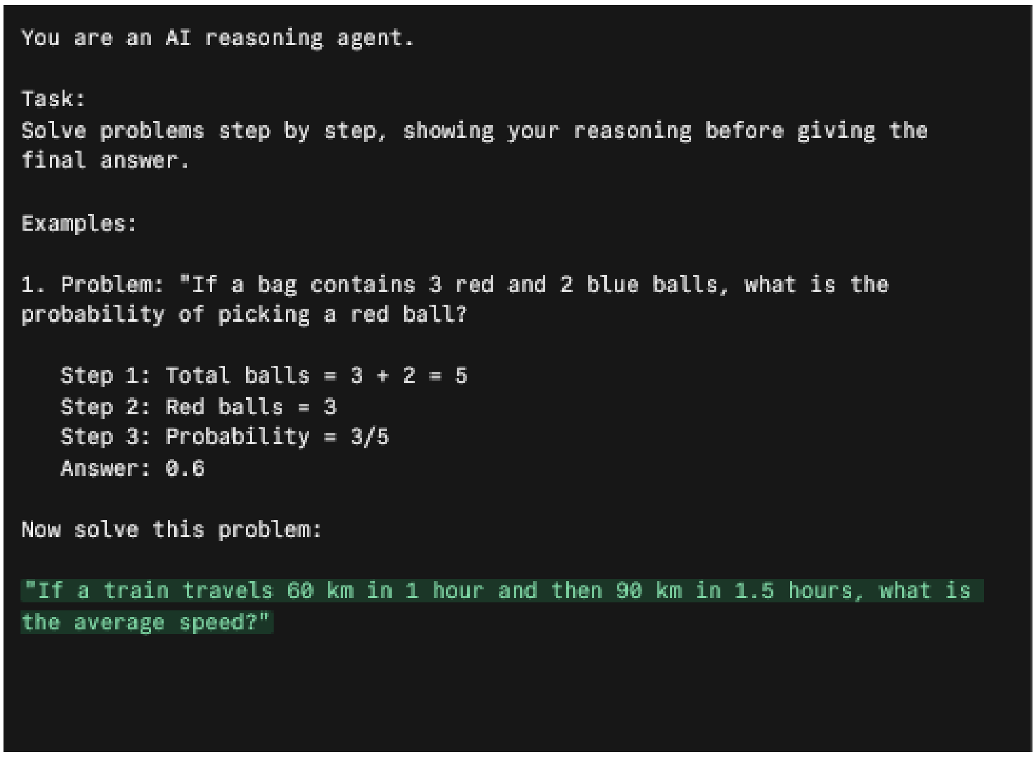


```
[
  {
    "output": "To find the average speed of the train, we need to calculate the total
              distance traveled and the total time taken, then divide the total distance
              by the total time.\n\nStep 1: Calculate the total distance traveled.\n-
              First segment: 60 km\n- Second segment: 90 km\n- Total distance = 60 km + 90
              km = 150 km\n\nStep 2: Calculate the total time taken.\n- First segment: 1
              hour\n- Second segment: 1.5 hours\n- Total time = 1 hour + 1.5 hours = 2.5
              hours\n\nStep 3: Calculate the average speed.\n- Average speed = Total
              distance / Total time\n- Average speed = 150 km / 2.5 hours = 60 km/
              h\n\nFinal Answer: The average speed of the train is 60 km/h."
  }
]
```

Figure IV:.18: Zero-Shot CoT output

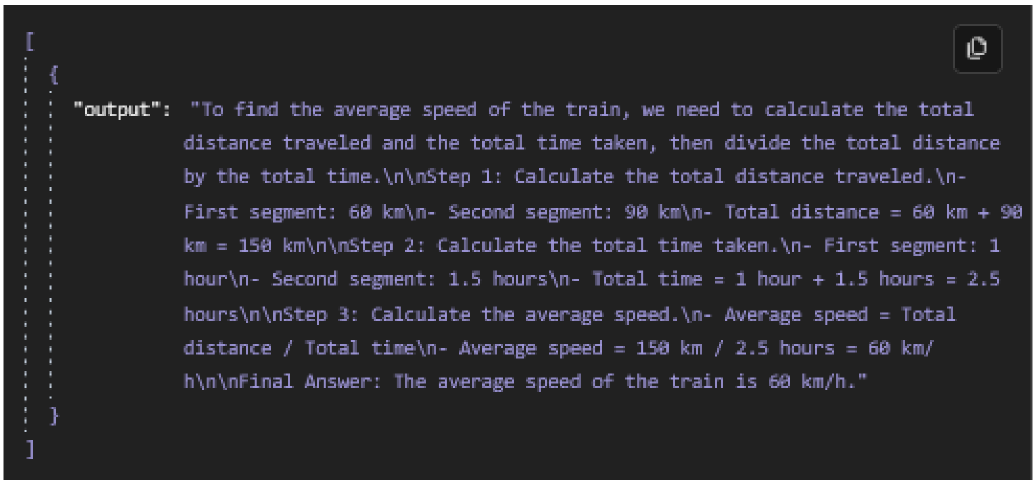
## 18).2 Few-Shot CoT

The few-shot CoT provides the LLM with **the exact steps to follow** when solving the problem, guiding its reasoning more precisely.



```
You are an AI reasoning agent.

Task:
Solve problems step by step, showing your reasoning before giving the
final answer.

Examples:

1. Problem: "If a bag contains 3 red and 2 blue balls, what is the
probability of picking a red ball?

   Step 1: Total balls = 3 + 2 = 5
   Step 2: Red balls = 3
   Step 3: Probability = 3/5
   Answer: 0.6

Now solve this problem:

"If a train travels 60 km in 1 hour and then 90 km in 1.5 hours, what is
the average speed?"
```

Figure IV:.19: Few-Shot CoT



```
[
  {
    "output": "To find the average speed of the train, we need to calculate the total
              distance traveled and the total time taken, then divide the total distance
              by the total time.\n\nStep 1: Calculate the total distance traveled.\n-
              First segment: 60 km\n- Second segment: 90 km\n- Total distance = 60 km + 90
              km = 150 km\n\nStep 2: Calculate the total time taken.\n- First segment: 1
              hour\n- Second segment: 1.5 hours\n- Total time = 1 hour + 1.5 hours = 2.5
              hours\n\nStep 3: Calculate the average speed.\n- Average speed = Total
              distance / Total time\n- Average speed = 150 km / 2.5 hours = 60 km/
              h\n\nFinal Answer: The average speed of the train is 60 km/h."
  }
]
```

Figure IV:.20: Few-Shot CoT output

## 19) AI Hallucination:

Sometimes, AI models like LLMs can give answers that sound believable but are actually wrong or made up. This is called **AI Hallucination.**

Hallucinations can happen for several reasons:

- The prompt is unclear or missing important details.
- The task requires facts or knowledge the model doesn't know.
- The model is asked to guess or fill in gaps.

Being aware of hallucinations is important because they can lead to wrong outputs or break your workflow. To reduce them, you can:

- Give clear instructions in your prompt.
- Provide examples (few-shot prompting) so the model knows what kind of answer to give.
- Use Chain of Thought (CoT) prompting to make the model think step by step before answering.

## 20) Model Temperature:

Temperature is a setting in language models that controls how creative or random the AI's outputs are.

- Low temperature (e.g., 0–0.3): The model is more deterministic and focused, giving predictable answers.
- Medium temperature (e.g., 0.4–0.7): Balances creativity and reliability.
- High temperature (e.g., 0.8–1.0): The model becomes more creative and varied, but can also hallucinate more.

## 20).1 Importance of Model Temperature:

**Reducing hallucinations:**Lowering the temperature makes the model less likely to make up facts or produce inconsistent outputs.
**Controlling creativity:** Higher temperatures are useful for storytelling, brainstorming, or open-ended tasks, but not for precise tasks like classification or calculations.

**Tip: For tasks that require accuracy, like text classification, sentiment analysis, or prompt-based reasoning, keep the temperature low. For creative writing or exploratory outputs, you can increase it.**

# Chapter V:

# RAG (Retrieval-Augmented Generation):

**RAG is a method where a language model retrieves relevant information from a database and then generates an answer based on that information.**

## 21) Vector:

**A vector is a numerical representation of meaning.**

The list can contain many numbers (for example 384, 768, or 1536 values), depending on the embedding model.

Computers **cannot understand words or language directly**. Instead, they work with **numbers.** By converting text into numbers, computers can compare meanings using mathematical operations.

For example, the words “cat” and “kitten” have very similar meanings, so their vectors are close to each other in vector space. This mathematical closeness is how computers recognize semantic similarity.

## 22) Embeddings:

**An embedding is the process of converting text into a vector.**

This process is performed by a trained model that transforms words, sentences, or entire paragraphs into numerical representations that capture their semantic meaning.

The goal of embeddings is not to **store the exact words**, but to **represent the meaning behind them**. As a result, different texts that express the same idea will have similar embeddings, even if they use different words.

For example, the sentences “What is a black hole?” and “Explain black holes” are phrased differently, but their embeddings will be close to each other because they share the same meaning.

Embeddings allow computers to perform semantic tasks such as similarity search, clustering, and information retrieval by comparing vectors mathematically rather than relying on keyword matching.

## 23) Chunks:

**A chunk is a small piece of text embedded separately.**

Instead of processing an entire document at once, the document is divided into multiple chunks, and each chunk is handled independently.

Chunking is necessary because large language models have context limits and because smaller pieces of text preserve meaning more accurately during retrieval.
**A chunk is a group of tokens, not a single token.**

## 24) Vector database:

A vector database is a system designed to store vectors and perform searches based on semantic meaning rather than exact matches.

Instead of looking for identical words, it finds vectors that are mathematically close to each other, which represent similar meanings.

Each stored vector is usually associated with a **text chunk**.

For example, when a user asks a question, the question is converted into a vector and compared against stored vectors in the database. The database then returns the most relevant text chunks based on similarity scores.

Pinecone, Qdrant, and Supabase are all examples of vector databases that store embeddings and perform semantic searches, allowing systems to find relevant information based on meaning rather than exact matches.

## 25) The Vector Retrieval Process:

### 25).1 Chunking:

The document is split into smaller, meaningful pieces called chunks. Each chunk contains multiple tokens to preserve semantic context.

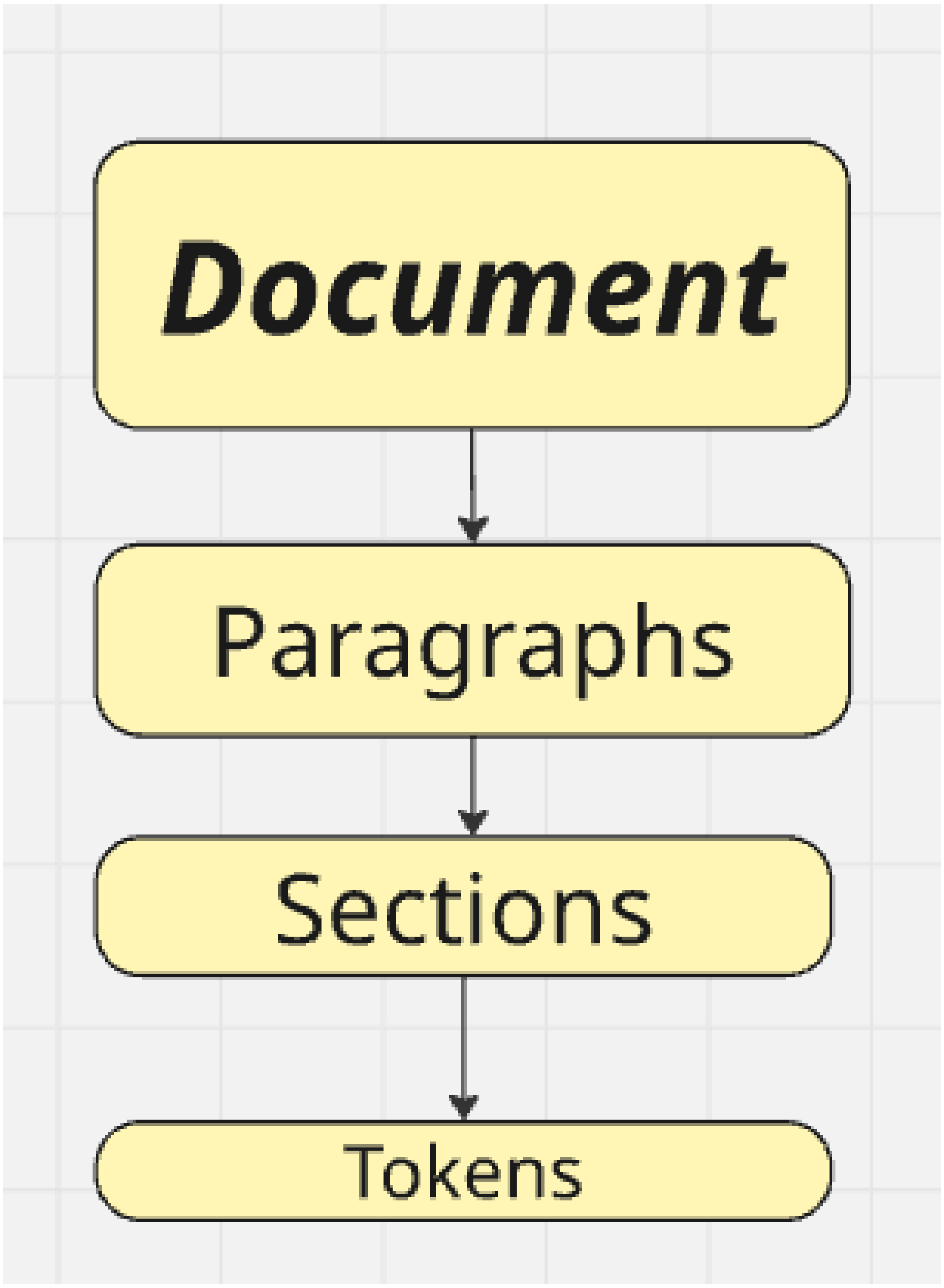


Figure V:.1: Process of transforming a document into chunks

### 25).2 Embedding:

Each chunk is converted into a vector using an embedding model, capturing the semantic meaning of the text.

### 25).3 Storing in a Vector Database:

The vectors, along with the chunk text and metadata (source, page, etc.), are stored in a vector database such as Pinecone, Qdrant, or Supabase.

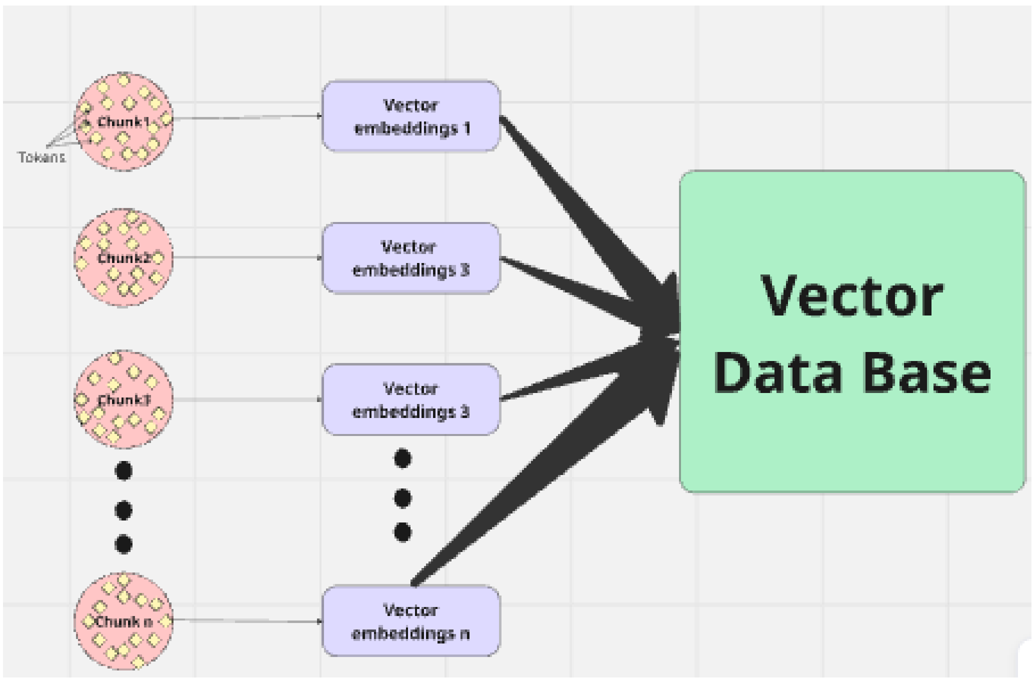


Figure V:.2: Process from chunks to vector embeddings to storing

## 25).4 User Query:

When a user asks a question, it is converted into an embedding vector using the same embedding model.

## 25).5 Similarity Search:

The vector database compares the query vector to stored chunk vectors and retrieves the top-K most relevant chunks based on similarity scores.

**The K chunks that are closest in meaning to the question.**

K = number of chunks you pick (like 3 or 5).

Most relevant = most related to the question.

Similarity score = a number showing how close the meaning is.

Example:
If K = 3 and the question is "What is a black hole?", the system picks the 3 chunks that talk most about black holes.

## 25).6 Providing Context to the LLM:

The retrieved chunks are passed as context to the LLM, which generates a grounded answer based on the information.

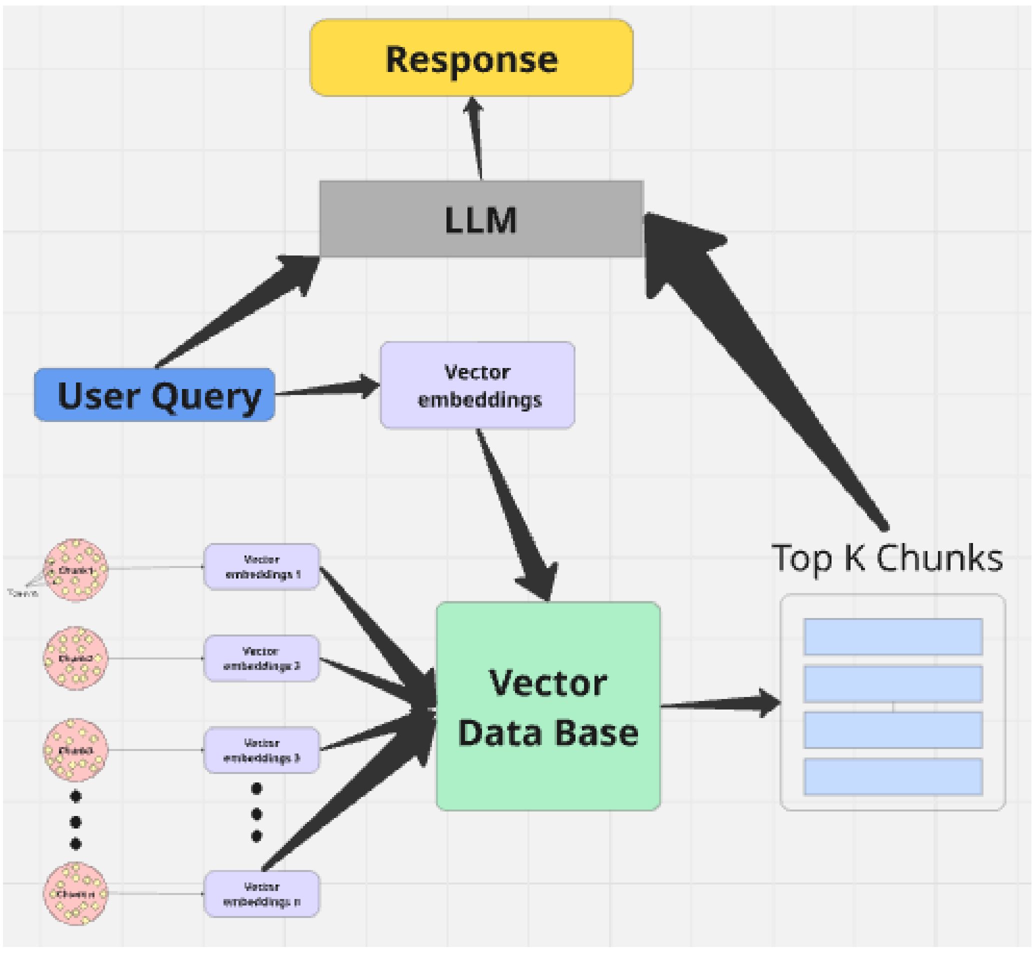


Figure V:.3: From user query to answering process

**Note: The provided document does not have to be text; it can also be an image, audio, or other types of data**

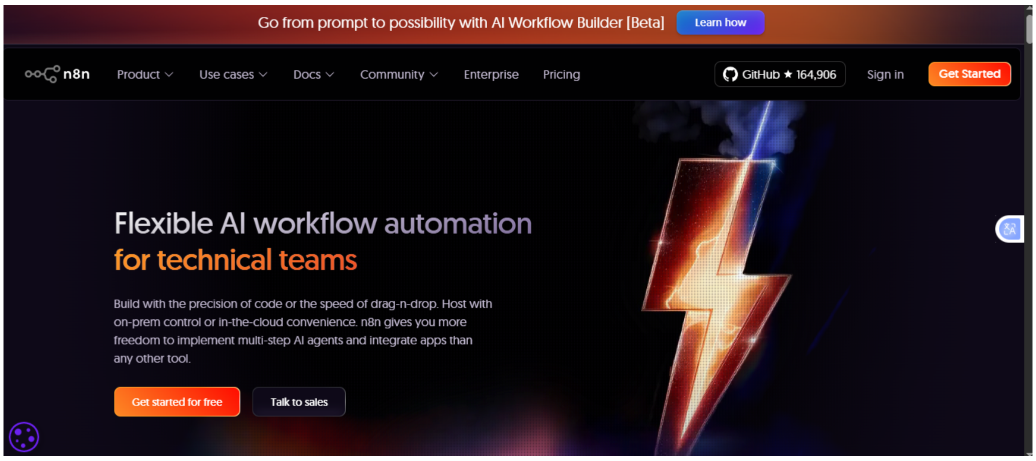
All of this process, from chunking and embedding documents, storing them in a vector database, retrieving relevant chunks for a query, and generating an answer with a language model, is called **Retrieval-Augmented Generation (RAG).**

# Chapter VI:

# N8N platform:

## 26) n8n Cloud

n8n Cloud is the hosted version of n8n, managed entirely by the n8n team. **Open your browser and visit n8n.io.** This will take you directly to the n8n cloud platform, where you can start with a free 14 day trial. You can also check the Pricing section to see whether a monthly subscription fits your budget. However, it is recommended to begin with a free option, at least for testing and learning purposes.



Go from prompt to possibility with AI Workflow Builder [Beta]

Flexible AI workflow automation
for technical teams

Build with the precision of code or the speed of drag-n-drop. Host with on-prem control or in-the-cloud convenience. n8n gives you more freedom to implement multi-step AI agents and integrate apps than any other tool.

Figure VI:.1: n8n.io

## 27) n8n Self Hosted:

Self hosted n8n means running n8n on your own machine or server.

# Chapter VII:

# DOCKER:

Figure VII:.1

**Docker is similar to a virtual machine**, but much lighter and faster. It allows you to build, run, and ship applications easily by packaging them with everything they need to work (code, libraries, and dependencies).

Docker works the same way on any machine, your laptop, another computer, or a server, so you don't face the common "it works on my machine" problem.

Instead of running full operating systems like virtual machines, Docker uses containers, which share the host system's OS while staying isolated from each other. This makes containers faster to start, more efficient, and less resource-hungry.

A single Docker host can run multiple containers, each created from a Docker image, allowing you to manage and deploy many applications at the same time in a clean and consistent way.

## 28) Docker vs VM in size and performance:

| **Feature** | **VM** | **DOCKER container** |
|---|---|---|
| **OS included** | **Full OS** | **Uses host OS kernel** |
| **Size** | **Heavy (GBs)** | **Lightweight (MBs)** |
| **Startup time** | **Minutes** | **Seconds** |
| **Resource usage** | **High** | **Low** |
| **Performance** | **Slower, extra layer** | **Near native** |

## 29) Docker component:

Docker is made of several core components that work together to build, run, and manage containers.

### 29).1 Docker Engine (The Core):

Docker Engine runs in the background and listens for Docker commands, it is the service that**builds images, runs containers, manages networks and volumes.**

**Docker Engine includes:**

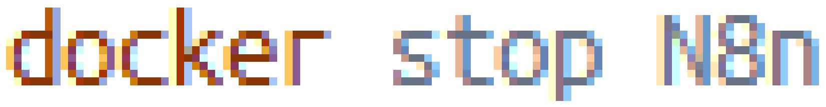
- Docker Daemon
- Docker API
- Docker CLI

### 29).2 Docker Daemon (dockerd):

The Docker Daemon is a background service responsible for creating images, starting and stopping containers, and managing storage, networks, and volumes. It performs these tasks based on commands received from the Docker CLI.

### 29).3 Docker CLI (Command Line Interface):

The Docker CLI is what you use to send instructions to the Docker Daemon.



```
docker stop N8n
```

Figure VII:.2: Example of a Docker CLI command

If a command starts with docker, it is a Docker CLI command.

## 29).4 Docker Images:

**A Docker image** It is a space that contains the application and its dependencies and is used to create containers.

**You create one image per application, but a single image can be used to create as many containers as you want**

**Before deleting an image, you need to delete all containers that were created from it; otherwise, the image will not be removed.**

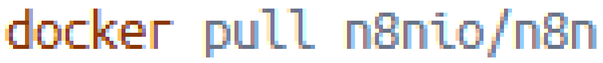
## 29).5 Docker Containers:

**A container** is an isolated system from the host, it is running instance of an image.

**Containers** Containers can be created and deleted easily and quickly, and they start and stop fast. However, if there is no image to start from, a container cannot be created.

## 29).6 Docker Registry (Docker Hub):

One of the most popular registry stores Docker images is **Docker Hub**, and from there you can: pull images and also push your own images.



```
docker pull n8nio/n8n
```

Figure VII:.3: Docker command to pull n8n image

## 29).7 Docker Volumes:

**Volumes** are used to store data outside containers because containers are temporary and can be removed easily, so Without volumes, data is lost when container stops.

**Volumes allow:** data persistence, backups, safe upgrades.

**Example use cases in n8n:** storing n8n workflows and credentials.

## 29).8 Docker Networks:

Docker networks allow containers to talk to each other, and stay isolated from the outside world.

**Example:** n8n container communicates with PostgreSQL container

## 29).9 Docker Compose:

Docker Compose is a tool to manage multiple containers.

Instead of long commands, you use one YAML file (docker-compose.yml)

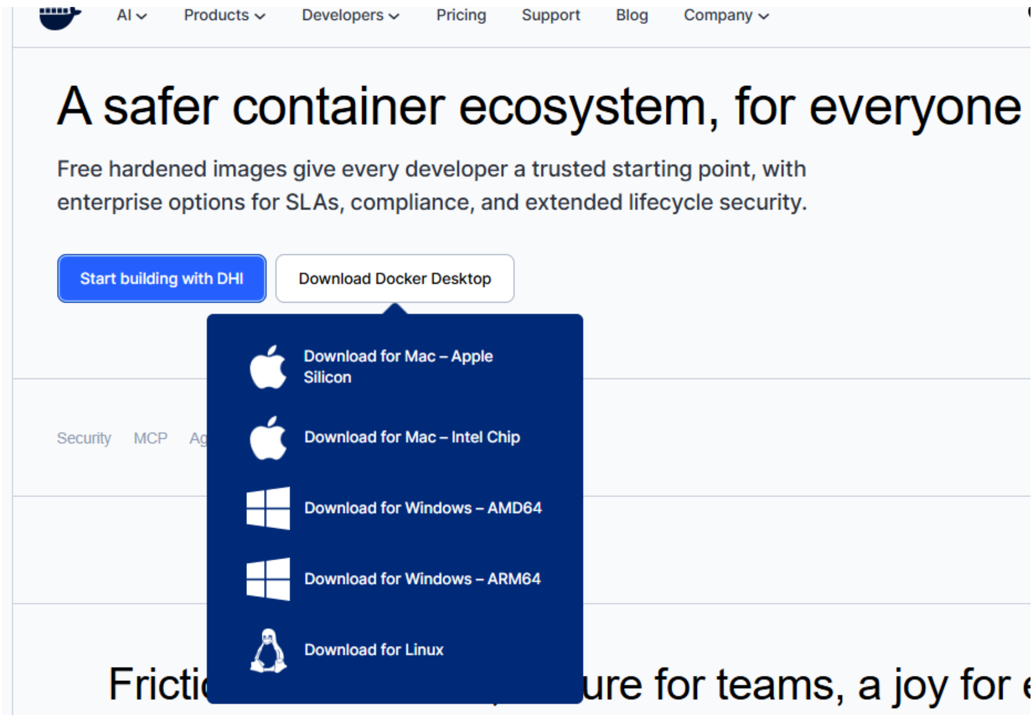
**We will mainly use Docker Desktop, so this subsection will be kept brief.**

# Chapter VIII:

# Docker for N8N:

## 30) Downloading Docker:

Go to docker.com and download Docker for your operating system.



A safer container ecosystem, for everyone

Free hardened images give every developer a trusted starting point, with enterprise options for SLAs, compliance, and extended lifecycle security.

Figure VIII:.1: Docker website

## 31) Creating n8n image:

After downloading Docker Desktop, go to the search bar at the top, search for n8n, select the one with 100M pulls, and pull the image.

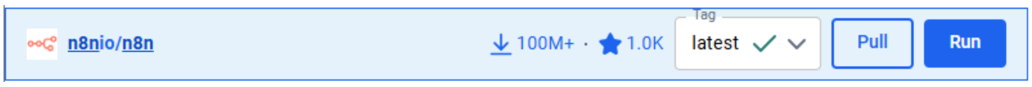

n8nio/n8n 100M+ · 1.0K
Tag latest

Figure VIII:.2: n8n Image

## 32) Create n8n container:

After pulling the image, go to Run, the Optional Settings. Now it's time to create your first container:

(1) Select a name for your container.

(2) Enter the port 5678.

(3) On your computer, create a folder to safely store all your credentials and workflows.

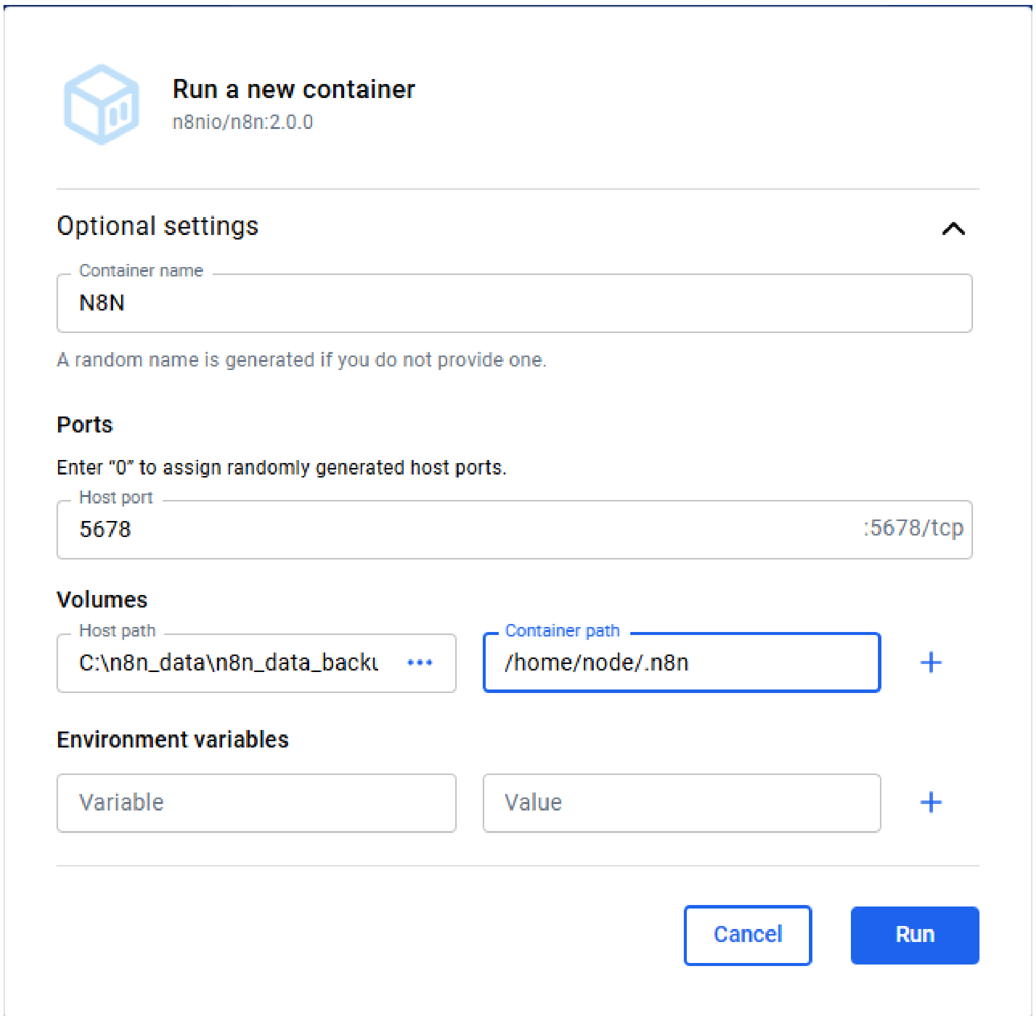
(4) In Docker, set this folder as the host path (volume).

(5) For the container path, use: /home/node/.n8n

(6) Environment variables are not needed for now.

(7) Click Run

Your container has now been created successfully! you can find it in Containers section.


Run a new container
n8nio/n8n:2.0.0

Optional settings

Container name: N8N

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port: 5678 :5678/tcp

Volumes

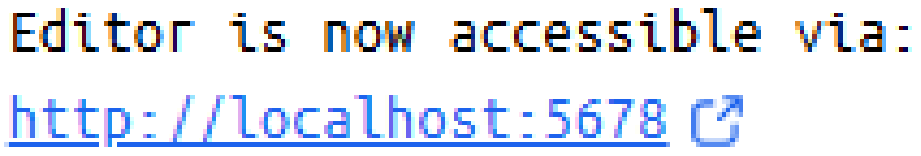
Host path: C:\n8n_data\n8n_data_backu
Container path: /home/node/.n8n

Environment variables

Variable | Value

Figure VIII:.3: Create n8n container

Go to Containers, find your container, and click Run to start it. Wait until your editor becomes accessible via localhost:5678. Once it is, click on the link, and you're now in n8n.

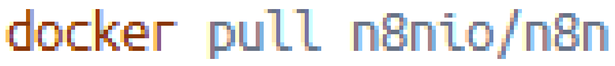


```
Editor is now accessible via:
http://localhost:5678
```

Figure VIII:.4: Localhost 5678

## 33) Work with Docker terminal:

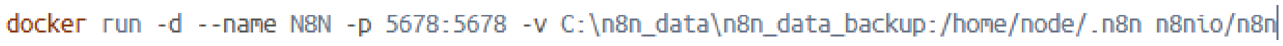
Up to this point, we've been using the GUI. Optionally, you can use the terminal (down on your left) to create the n8n image:



```
docker pull n8nio/n8n
```

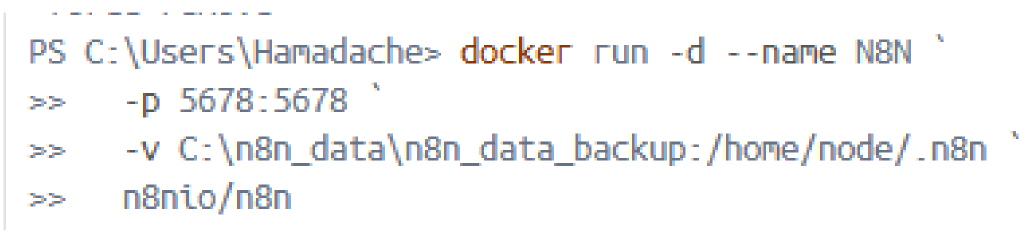
Figure VIII:.5: create n8n image

Run one of the following commands to create the container:



```
docker run -d --name N8N -p 5678:5678 -v C:\n8n_data\n8n_data_backup:/home/node/.n8n n8nio/n8n
```

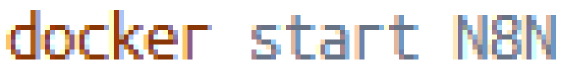
Figure VIII:.6: Create n8n container(Command 1)



```
PS C:\Users\Hamadache> docker run -d --name N8N `
>>    -p 5678:5678 `
>>    -v C:\n8n_data\n8n_data_backup:/home/node/.n8n `
>>    n8nio/n8n
```

Figure VIII:.7: Create n8n container(Command 2)

Run the container:

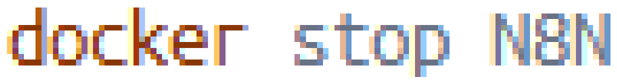


```
docker start N8N
```

Figure VIII:.8: Running the container

**Wait for a minute before tapping** `http://localhost:5678/` **on your browser to access your n8n session**

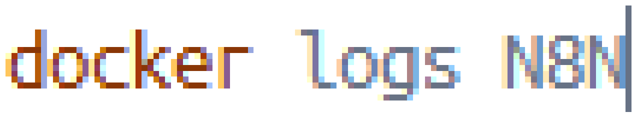
Stop the container:



```
docker stop N8N
```

Figure VIII:.9: Running the container

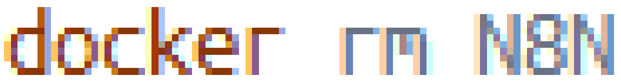
View the logs:



```
docker logs N8N
```

Figure VIII:.10: View the logs

Remove the container:



```
docker rm N8N
```

Figure VIII:.11: Remove the container

### 33).1 Decoding the Docker command:

The figures (VIII:.6 and VIII:.7) displays the command to create a container, which we will decode in this subsection.

- docker run -d: "d" for detached mode, this command mean running the container in the background ( Use -"it" instead of "-d" if you want the container to depend on the terminal, meaning it will stop when the terminal is closed.)
- –name N8N: gives your container a name.
- -p 5678:5678: "p" for port, container port 5678 to your computer port 5678.
- `-v C:\n8n_data\n8n_data_backup:/home/node/.n8n`: "v" for volume, mounts your local folder (`C:\n8n_data\n8n_data_backup`) to the container for persistent data (`/home/node/.n8n`).
- n8nio/n8n: The image to run.

**We are now running n8n using Docker on localhost.**

## 34) Updating n8n in Docker:

To update your n8n instance in Docker, first go to the Images section and search for n8n (as described in section 31)). Then, go to Tag and select the newest version, or simply choose the version number you want—and pull it.

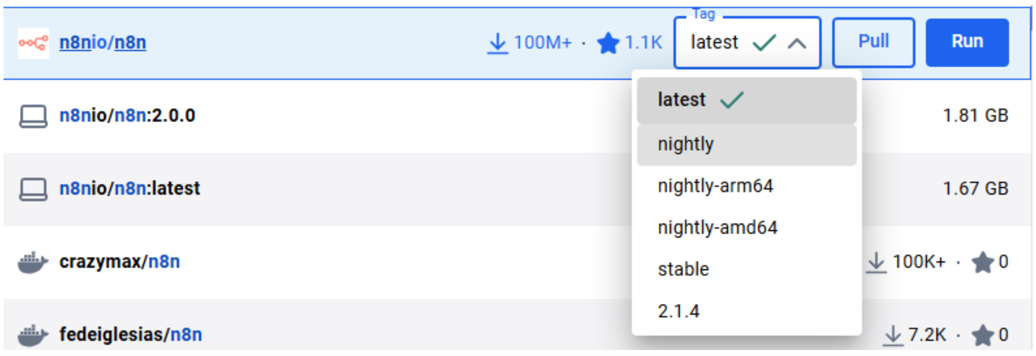


Figure VIII:.12: The newest image

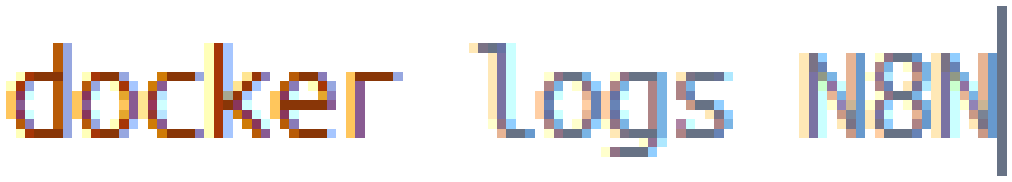
Before running the new image, delete the old container (don't worry—everything is saved in the volume, not the container). Then, run the newest image to create your updated container (follow the same steps in 32)). Your updated instance will now be fully set up. **Note: You can delete the old image after deleting its container.**

**You can choose not to delete the container, as long as you change the container name and the port so the containers do not conflict with each other.**

## 35) Docker extra:

In the Containers section select a container a container, you will see logs, inspect, bind mounts, exc, files, stat:

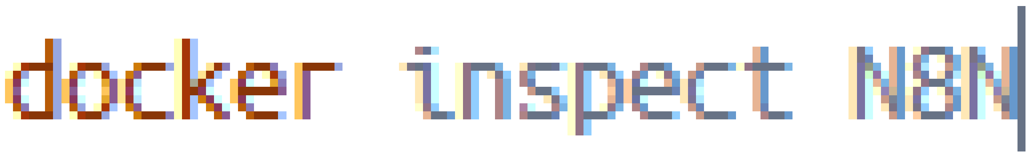
**1.logs:**
Displays the real time output generated by the container. This is useful for debugging, checking errors, and confirming that services (such as n8n) are running correctly.



```
docker logs N8N
```

Figure VIII:.13: Logs command in docker terminal

**Inspect:**
Shows detailed technical information about the container in JSON format, including configuration, environment variables, network settings, ports, and mounted volumes.



```
docker inspect N8N
```

Figure VIII:.14: Inspect command in docker terminal

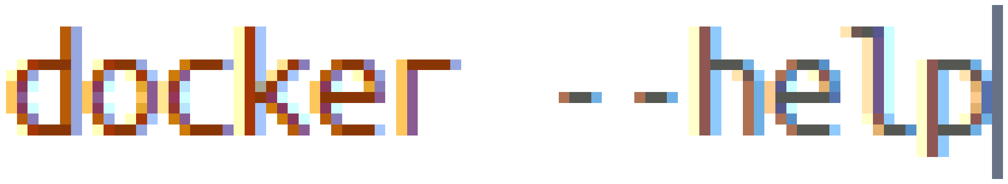
**Bind Mounts:**
Lists the local folders (volumes) that are mounted into the container. This is where persistent data is stored, ensuring your workflows and credentials remain safe even if the container is deleted.

**Exec:**
Allows you to open a terminal inside the running container. From here, you can execute commands directly inside the container environment.

**Files:**
Lets you browse the container's file system and view or download files inside it.

**Stats:**
Displays real time resource usage such as CPU, memory, network, and disk I/O, helping you monitor the container's performance.

**Note: You can use the Docker command `docker --help` to view all available Docker commands along with a brief description of what each command does. This is useful for discovering new commands and understanding how to use Docker directly from the terminal.**



```
docker --help
```

Figure VIII:.15: Docker help

# Chapter IX:

# NGROK:

## 36) Introduction:

Now that we have an n8n instance running in Docker, we can start working with it. However, this instance is strictly local (localhost). If we want to use webhooks or any node that requires communication with a third-party service, the workflow will not work, because external services cannot reach a local instance.

To solve this, the instance needs to be accessible from the internet. For beginners(Testing), we introduce **"ngrok"**, **a tunneling tool** that exposes your local instance to the web and provides **a public domain for free.**

## 37) Ngrok:

**Ngrok**is a tool that creates a secure tunnel from the public internet to your local machine (that means, it allows people (or services) outside your network to access a service running on your computer.)

## 38) Ngrok Key Features:

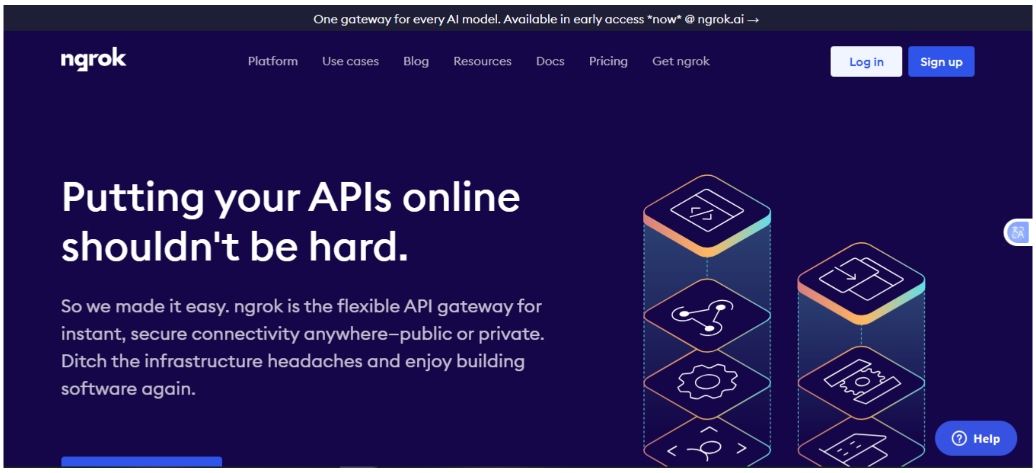
- Secure tunnels with HTTPS.
- Temporary public URLs that point to local services.
- Works on Windows, Mac, Linux.
- It allows you to connect the domain it provides directly to your Docker container.

## 39) Limitations of Ngrok:

- Ngrok is designed for development and testing, not for production systems.
- Your workflow depends on ngrok being available: If ngrok is down or your tunnel disconnects, webhooks stop working.
- It is not reliable enough for workflows that must run 24/7: (Ngrok stops when the internet connection is stopped or your computer has been shot down)
- Requests go through ngrok's servers, which can introduce latency:this may slow down webhook responses.

## 40) Downloading Ngok:

In your browser tap ngrok.com.


One gateway for every AI model. Available in early access *now* @ ngrok.ai →
ngrok
Platform
Use cases
Blog
Resources
Docs
Pricing
Get ngrok
Log in
Sign up
Help

Putting your APIs online shouldn't be hard.

So we made it easy. ngrok is the flexible API gateway for instant, secure connectivity anywhere—public or private. Ditch the infrastructure headaches and enjoy building software again.

Figure IX:.1: Ngrok Website

Go to "Get ngrok" on top left of the website, and here u can downoald ngrok based on your system (Windows, Mac or Linux).

## 41) Windows:

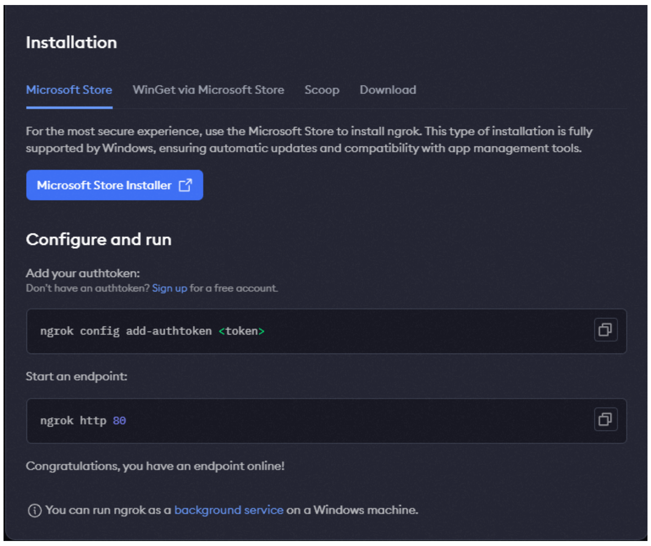
Follow these instructions:

(1) Install ngrok from the Microsoft Store (It is advisable to install from the Microsoft store for Windows users.).

(2) Sign up for an ngrok account and obtain your authentication token.

(3) Open your terminal and run the following command: `ngrok config add-authtoken <token>`.

(4) Start the tunnel by running: `ngrok http 80`.



Installation

For the most secure experience, use the Microsoft Store to install ngrok. This type of installation is fully supported by Windows, ensuring automatic updates and compatibility with app management tools.

Configure and run

Add your authtoken:
Don't have an authtoken? Sign up for a free account.

```
ngrok config add-authtoken <token>
```

Start an endpoint:

```
ngrok http 80
```

Congratulations, you have an endpoint online!

You can run ngrok as a background service on a Windows machine.

Figure IX:.2: Windows instructions

## 42) Mac:

Follow these instructions:

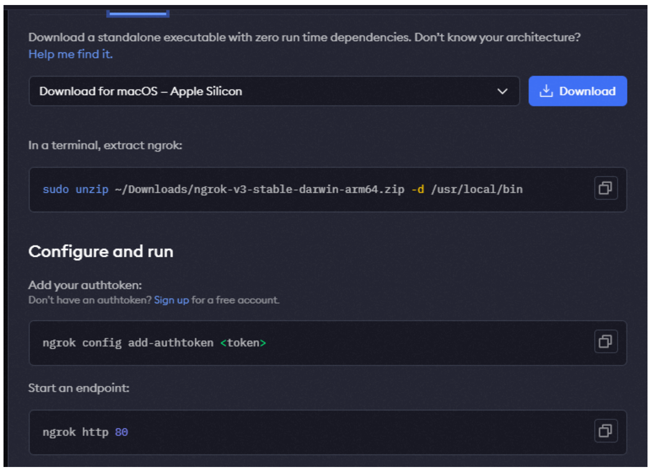
(1) Download ngrok from the website using the link provided.

(2) In the terminal, extract ngrok using the following command:
`sudo unzip ~/Downloads/ngrok-v3-stable-darwin-arm64.zip -d /usr/local/bin`

(3) Sign up for an ngrok account and obtain your authentication token.

(4) Open the terminal and run the following command:
`ngrok config add-authtoken <token>`

(5) Start the tunnel by running:
`ngrok http 80`



Download a standalone executable with zero run time dependencies. Don't know your architecture? Help me find it.

In a terminal, extract ngrok:

```
sudo unzip ~/Downloads/ngrok-v3-stable-darwin-arm64.zip -d /usr/local/bin
```

Configure and run

Add your authtoken:
Don't have an authtoken? Sign up for a free account.

```
ngrok config add-authtoken <token>
```

Start an endpoint:

```
ngrok http 80
```

Figure IX:.3: Mac instructions

## 43) Linux:

(1) Download ngrok from the website using the provided link, or install it using the `apt` command.

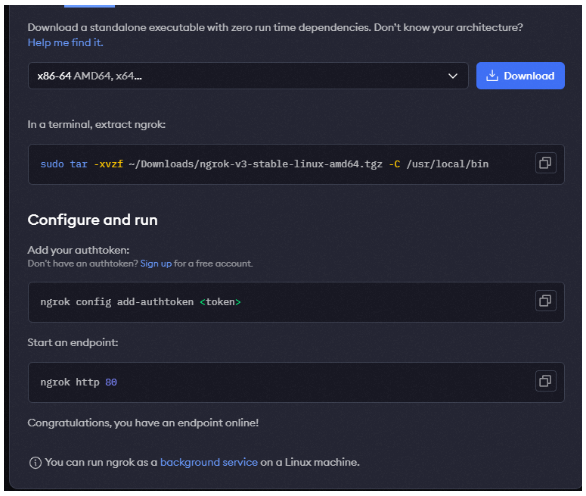
(2) If you choose to download ngrok from the website, open the terminal and extract it using the following command:
`sudo tar -xvzf ~/Downloads/ngrok-v3-stable-linux-amd64.tgz -C /usr/local/bin`

(3) If you installed ngrok via `apt`, you may skip this step and proceed to the next one.

(4) Create an ngrok account and obtain your authentication token.

(5) Open the terminal and run the following command to add the token:
`ngrok config add-authtoken <token>`

(6) Start the tunnel by running the following command:
`ngrok http 80`



Download a standalone executable with zero run time dependencies. Don't know your architecture?
Help me find it.

In a terminal, extract ngrok:

```
sudo tar -xvzf ~/Downloads/ngrok-v3-stable-linux-amd64.tgz -C /usr/local/bin
```

Configure and run

Add your authtoken:
Don't have an authtoken? Sign up for a free account.

```
ngrok config add-authtoken <token>
```

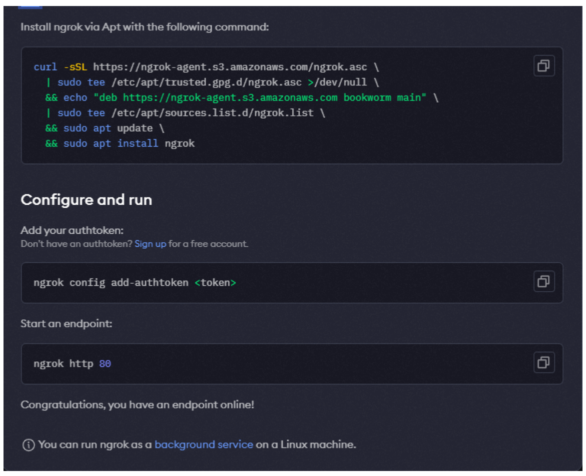
Start an endpoint:

```
ngrok http 80
```

Congratulations, you have an endpoint online!

ⓘ You can run ngrok as a background service on a Linux machine.

(a) Linux direct download instructions



Install ngrok via Apt with the following command:

```
curl -sSL https://ngrok-agent.s3.amazonaws.com/ngrok.asc \
  | sudo tee /etc/apt/trusted.gpg.d/ngrok.asc >/dev/null \
  && echo "deb https://ngrok-agent.s3.amazonaws.com bookworm main" \
  | sudo tee /etc/apt/sources.list.d/ngrok.list \
  && sudo apt update \
  && sudo apt install ngrok
```

Configure and run

Add your authtoken:
Don't have an authtoken? Sign up for a free account.

```
ngrok config add-authtoken <token>
```

Start an endpoint:

```
ngrok http 80
```

Congratulations, you have an endpoint online!

ⓘ You can run ngrok as a background service on a Linux machine.

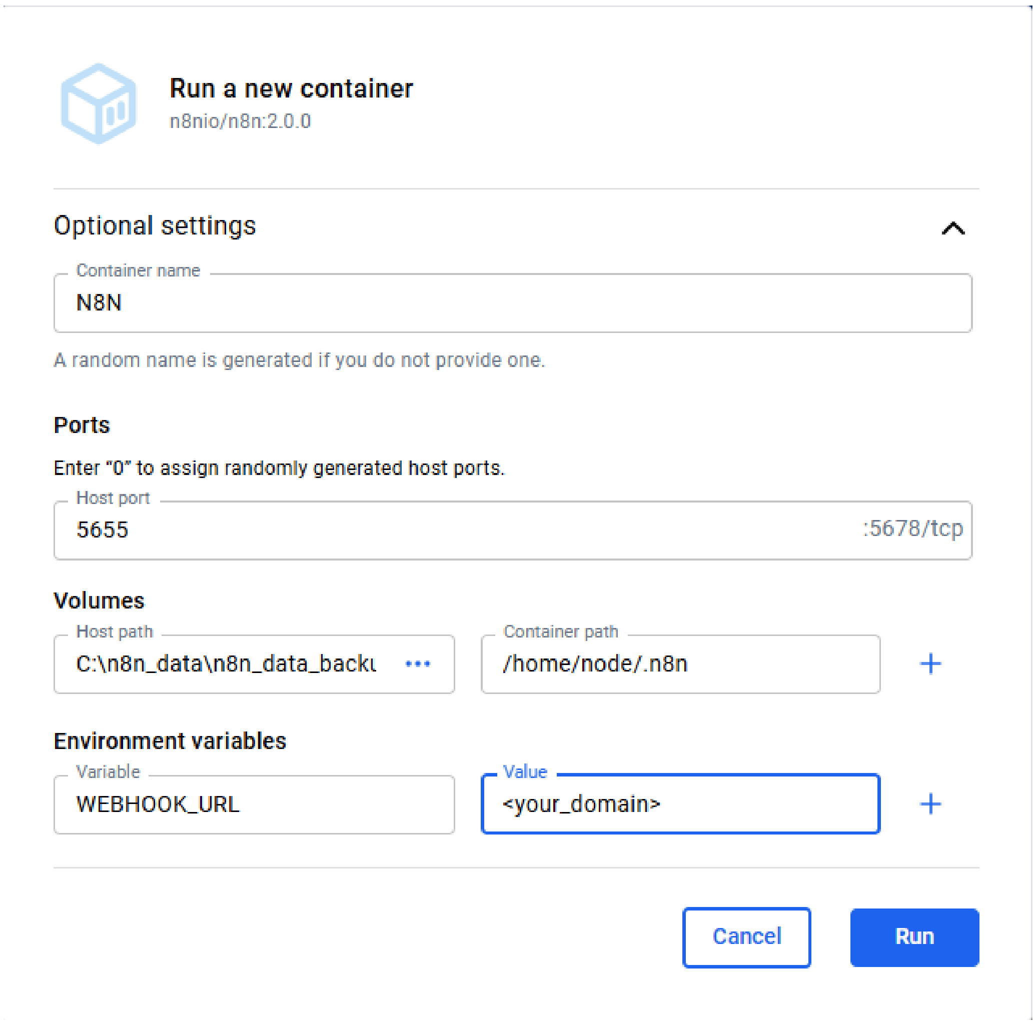
(b) Linux via Apt instructions

## 44) Your ngrok domain:

After running the command `ngrok http 80`, ngrok will generate a **public domain** for you. **Do not share this domain with anyone**.
**This domain will serve as your webhook URL and will be used later when working with Docker.**

## 45) Docker with ngrok:

Create a new container (see Section 32)), ensuring that it has a **name different from your local host container**. Change the port to **any value other than 5678**. Keep the same volumes so that your workflows and credentials are not lost. Finally, in the environment variables, add `WEBHOOK_URL` as the variable name and set its value to the domain provided by `ngrok`.



Run a new container
n8nio/n8n:2.0.0

Optional settings

Container name: N8N

A random name is generated if you do not provide one.

Ports

Enter "0" to assign randomly generated host ports.

Host port: 5655 :5678/tcp

Volumes

Host path: C:\n8n_data\n8n_data_backu

Container path: /home/node/.n8n

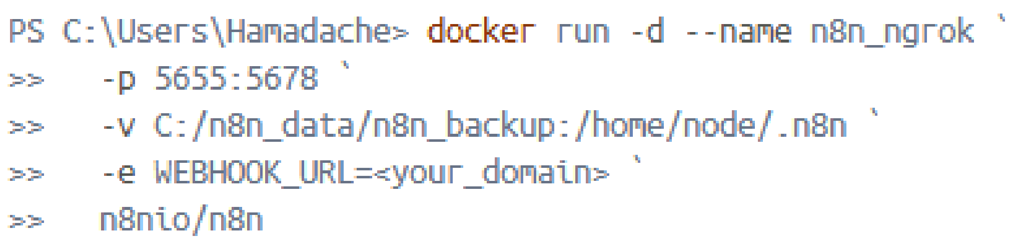
Environment variables

Variable: WEBHOOK_URL

Value: <your_domain>

Figure IX:.5: Create your ngrok container in the GUI

Perform the same steps using the Docker terminal with the following command:

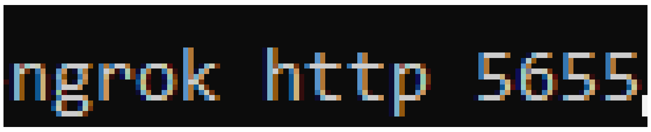


```
PS C:\Users\Hamadache> docker run -d --name n8n_ngrok `
>>    -p 5655:5678 `
>>    -v C:/n8n_data/n8n_backup:/home/node/.n8n `
>>    -e WEBHOOK_URL=<your_domain> `
>>    n8nio/n8n
```

Figure IX:.6: Create your ngrok container in docker terminal

Run the container, then open your terminal (or the ngrok terminal) and execute the following command: `ngrok http 5655`.

**Note: 5655 is the port; if you set a different port, replace 5655 with your chosen port.**



```
ngrok http 5655
```

Figure IX:.7: Running ngrok

Now, open your browser and paste your ngrok domain. Your n8n instance is now running in the cloud.

# Chapter X:

# Production and deployment:

## 46) Deployment:

**Deployment** is the process of taking your n8n workflows from your local/dev setup and making them run on a server or cloud instance so they can work 24/7.

Setting up:
Environment variables, Credentials, Database, Webhook URLs.
**Making sure workflows start automatically and don't stop when you close your computer Examples of deployment in n8n:**

- Deploy n8n using Docker on a VPS.
- Deploy n8n on n8n Cloud.
- Deploy n8n on Hostinger / DigitalOcean / AWS.
- Deploy n8n behind Nginx + HTTPS.

### 46).1 n8n Deployment: Cloud vs Self Hosted

| **Aspect** | **Cloud** | **Self Hosted** |
|---|---|---|
| **Hosting** | **Managed by n8n team** | **Managed by you** |
| **Setup** | **Fast** | **Needs setup** |
| **Infrastructure** | **Fully handled by n8n** | **You manage server, OS, updates** |
| **Database** | **Managed by n8n** | **PostgreSQL recommended** |
| **Security** | **Managed by n8n** | **You are responsible** |
| **Cost** | **Monthly subscription** | **Server cost (often cheaper long-term)** |

## 47) Production:

**Production** is the live, real environment where your workflows are actively running and handling real data.

**In production:**

- Webhooks receive real user data.
- APIs are called with real credentials.
- Errors matter (they can break things)
- **Workflows are: Active (published), stable, and monitored.**

It is important to check for: Reliability, Performance, Security, Logs & error handling.

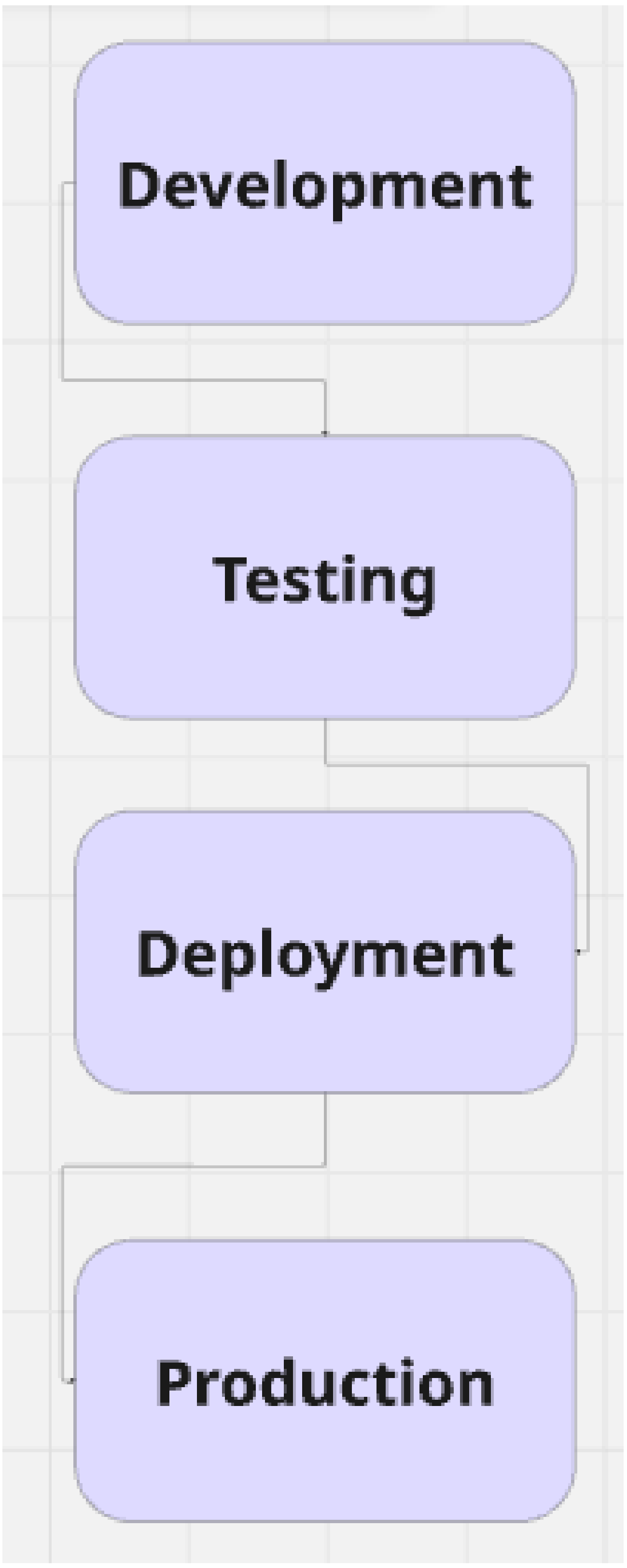


Figure X:.1: Workflow Lifecycle

## 48) Sending the workflow in JSON format:

In production, sending JSON is a commitment. Every client that consumes your data expects a specific structure, specific fields, and usable values.
Even if a workflow runs successfully, the JSON it produces can still be incomplete or incorrect.
n8n does not automatically validate JSON, which means validation must be done intentionally inside the workflow. A production ready workflow always verifies its JSON before sending it to any external system.

**Sending incorrect JSON is worse than sending no data at all.**

Most failures in production are not caused by crashes, but by silent data errors.

A workflow may execute without errors while still producing unusable output.
Once incorrect JSON is delivered, the failure moves to the client side, where it becomes harder to detect and fix.

### 48).1 Real Production Failure Examples:

The following issues are common in n8n workflows:

- A required field is missing: a field that should output an id for example, but the workflow sends null or omits the field entirely.
- Wrong data type: a numeric value is sent as a string ("42" instead of 42). The client's API may accept the request but processes the value incorrectly.
- Empty arrays or objects: a workflow sends an empty list where at least one item is required, causing downstream logic to fail.
- Accidental field renaming: a small change like `orderId` instead of `order_id` silently breaks the integration.
- Partial data delivery: some fields are present, others are missing, resulting in inconsistent client behavior.

### 48).2 Handling errors:

**JSON validation must happen before:**

1. Sending an HTTP request.
2. Returning a webhook response.
3. Delivering data to a client.

**Set nodes should be used to explicitly define the final JSON structure. If a field is not defined, it should not exist in the output.**

**IF nodes should be used to verify:**
1- Required fields exist.

2- Values are not empty.

3- Status values are valid.

4- Arrays contain expected data.

**If validation fails, the workflow should stop immediately. When validation fails:**

1. Stop the workflow.

2- Return a clear error response.

3- Log the failure for later review.

**Error handling should make problems visible by:**

Recording failed executions.

Returning meaningful error messages.

Allowing fast debugging and correction.

# Acknowledgement

This book was never meant to teach you everything.
If you reached this point, you no longer see automation as a black box. You understand what runs, where it runs, and why it fails. You know how to think before you build, and how to recover when something breaks.
I turned my ideas into this book, every error I faced, every question I asked myself along the way. Some problems took me hours to fix, and some questions had no one to answer them.
You may have questions that are not covered here, and you will likely encounter errors I never faced. Still, by reading this, you will learn the fundamentals, and, more importantly, you will learn how to think through problems and fix things on your own.
Do not copy blindly. Do not automate what you do not understand. And do not wait for permission to experiment.
You now have enough knowledge to continue alone.
Build slowly. Break things safely. Learn deeply.
And when your system finally works, remember: it was never the tool that made it possible, it was you.

## This book ends here.

## Your work does not.