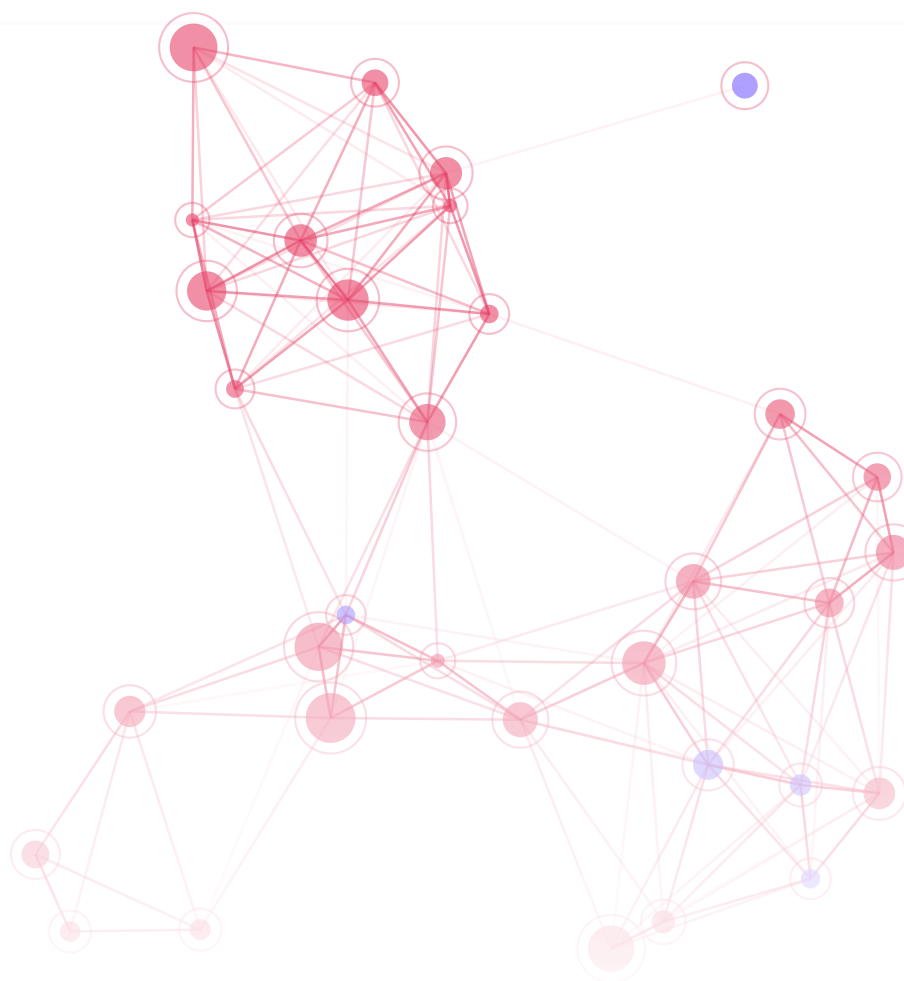




A BEGINNER'S MANUAL

CLAUDE EDITION

n8n

Automate your work and build AI agents, visually, one node at a time, from your first click to your own server

n8n

Automate your work and build AI agents, visually, one node at a time, from your first click to your own server

CLAUDE EDITION Expanded edition written by Claude (Anthropic's AI) in Claude Code and published in GuidesForEverything PR #4.

n8n (say "n-eight-n," short for "nodemation") is a visual automation tool. You connect building blocks, called **nodes**, on a canvas to make **workflows** that run on their own: "every morning, check the weather and email me," "when someone fills in my form, add them to a spreadsheet and send a welcome message," or "when a support email arrives, have an AI draft a reply."

n8n connects to more than a thousand apps, has powerful AI features built in, and, unlike most automation tools, you can run it on your own computer or server for free. This manual assumes no technical background. It takes you from your first click to data, expressions and code, AI agents with memory and knowledge, MCP, a dozen complete step-by-step recipes, and running your own reliable, secure n8n server.

HOW TO READ THIS MANUAL

Tip — a shortcut or good habit.

Warning — easy to get wrong.

\$ command — text you type, exactly as shown.

Note — useful background.

Danger — can lose data, money, or access.

Settings > Privacy — a menu path to click.

New words appear in **bold** the first time and are collected in the Glossary at the back.

Contents

01	Meet n8n	10
1.1	What is workflow automation?	10
1.2	What can you automate?	11
1.3	How n8n compares	11
1.4	What "fair-code" means for you	12
1.5	Cloud or self-hosted?	12
1.6	How this manual is organised	12
02	Key Ideas in Five Minutes	14
2.1	Items: the most important idea	15
2.2	Nodes come in a few families	15
2.3	Draft versus published	16
2.4	Three kinds of run	16
03	Cloud or Self-Hosted? Making the Choice	17
3.1	The two options in plain English	17
3.2	Side by side	18
3.3	Choose n8n Cloud if...	18
3.4	Choose self-hosting if...	18
3.5	A quick decision guide	19
3.6	The self-hosted editions	19
3.7	Switching later	20
04	Get n8n Running	21
4.1	Option A: n8n Cloud (recommended)	21
4.2	Option B: self-host on your own computer	22
4.3	Create the owner account	24
4.4	Turn on two-factor authentication	25
05	Self-Hosting on a Server	26
5.1	The pieces, explained	27
5.2	Step 1: rent a server	27
5.3	Step 2: point your domain at it	28
5.4	Step 3: install Docker and open the firewall	28
5.5	Step 4: create the project folder and secrets	28
5.6	Step 5: tell Caddy about your domain	29
5.7	Step 6: write the Compose file	30
5.8	What those settings do	31
5.9	Step 7: start everything	32

5.10	Everyday commands	32
5.11	Other ways to self-host	32
5.12	If it doesn't work	33
06	A Tour of the Editor	34
6.1	The Overview page	34
6.2	The main areas	35
6.3	Adding nodes	35
6.4	Connecting nodes	35
6.5	The node details view	35
6.6	Fixed values and expressions	36
6.7	Sticky notes and groups	36
6.8	The command bar	36
6.9	Keyboard shortcuts worth learning	36
6.10	The Executions tab	37
6.11	Settings you'll visit	37
07	Your First Workflow: A Morning Weather Email	38
7.1	Finding your latitude and longitude	40
7.2	What just happened, node by node	40
7.3	Add a decision	40
7.4	Check that it really ran	41
7.5	Make it yours	41
08	How Data Flows: Items, JSON and Binary	42
8.1	JSON in five minutes	42
8.2	The shape of n8n data	43
8.3	How nodes process many items	44
8.4	Inspecting data	44
8.5	Files: binary data	45
8.6	Splitting and combining items	46
8.7	Item linking	46
8.8	Pinning and mocking data	46
09	Expressions: Little Formulas, Big Power	48
9.1	Mapping by dragging	48
9.2	Expression mode	48
9.3	Expressions cheat sheet	49
9.4	Reaching your data	49
9.5	Built-in variables	50
9.6	Helper functions	50
9.7	Functions for text, numbers and lists	50
9.8	Dates and times with Luxon	51
9.9	Expressions or nodes?	52
9.10	Worked example: a friendly order summary	52

9.11	Common expression errors	52
10	The Code Node: JavaScript and Python	54
10.1	When to use code	54
10.2	The two modes	55
10.3	What goes in and what comes out	55
10.4	Five JavaScript recipes	56
10.5	Built-ins you can use in code	57
10.6	Python	57
10.7	Libraries in JavaScript	58
10.8	Keeping items linked	58
10.9	Let AI write it	58
10.10	Common Code node errors	59
11	Core Nodes: The Toolkit You'll Use Every Day	60
11.1	Edit Fields (Set)	60
11.2	Filter	61
11.3	If	61
11.4	Switch	61
11.5	Merge	61
11.6	Loop Over Items (Split in Batches)	62
11.7	HTTP Request	62
11.8	Wait	63
11.9	Everyday helpers	63
12	Triggers: How Workflows Start	65
12.1	Schedule Trigger	66
12.2	Webhook	67
12.3	Test URLs versus production URLs	67
12.4	n8n Form Trigger	68
12.5	Chat Trigger	68
12.6	App triggers: instant or polling?	69
12.7	Other triggers	69
13	Connecting Your Apps: Credentials and OAuth	70
13.1	Credentials	70
13.2	How OAuth works, in plain English	71
13.3	Managed versus custom OAuth	71
13.4	Walkthrough: Google OAuth on self-hosted n8n	71
13.5	Restricting where a credential can be used	72
13.6	Fixed and end-user credentials	72
13.7	When there's no node: HTTP Request	72
13.8	Sharing credentials	73
13.9	Security habits	73

14	Sub-workflows: Build Once, Reuse Everywhere	74
14.1	Why bother?	74
14.2	Create the sub-workflow	74
14.3	Call it from another workflow	75
14.4	How data passes between workflows	76
14.5	Convert part of a workflow	76
14.6	Ideas for your library	76
15	Data Tables: A Spreadsheet Inside n8n	77
15.1	Good uses	77
15.2	Create a table	77
15.3	The Data Table node	78
15.4	Worked example: skip what you've seen	78
15.5	Limits	79
15.6	Data tables, variables or a real database?	79
16	Saving, Publishing and Versions	80
16.1	Autosave and drafts	80
16.2	Publishing	80
16.3	Workflow history	81
16.4	Exporting and importing	82
16.5	Staying organised	82
16.6	Workflow settings worth knowing	82
16.7	Git and environments (advanced)	83
17	Handling Errors Gracefully	84
17.1	Retry On Fail	85
17.2	On Error	85
17.3	Error workflows	85
17.4	Stop and Error	86
17.5	Patterns for common problems	86
18	Debugging Executions	87
18.1	The Executions list	87
18.2	Reading a failed execution	88
18.3	Debug in editor	88
18.4	Retrying	88
18.5	Custom execution data	88
18.6	A debugging routine	89
18.7	Decoding common error types	89
19	AI in n8n: The Basics	91
19.1	A few AI words	91
19.2	AI steps in a workflow	92
19.3	Connecting a chat model	92

19.4	Getting structured answers	93
19.5	Writing good prompts	93
19.6	Running models locally	94
19.7	Let AI build workflows for you	94
20	The AI Agent Node in Depth	95
20.1	How an agent works	95
20.2	The AI Agent node	95
20.3	Writing the system message	97
20.4	Agent options	97
20.5	Memory	97
20.6	Tools	98
20.7	Human review for risky tools	99
20.8	Guardrails	99
20.9	Keeping agents dependable	99
20.10	Agents (first-class, in preview)	99
21	Knowledge for Your AI: Vector Stores and RAG	100
21.1	The idea in plain English	101
21.2	The two halves of RAG	101
21.3	Build a knowledge base	101
21.4	Ask questions with an agent	102
21.5	Choosing a vector store	102
21.6	Keeping it fresh	103
21.7	Improving answers	103
22	MCP: n8n and Other AI Apps	104
22.1	Using MCP servers in your agents	104
22.2	Turn a workflow into an MCP server	105
22.3	Instance-level MCP: let AI build for you	106
23	Recipes: Everyday Automations	107
23.1	Recipe 1: RSS-to-Slack digest	107
23.2	Recipe 2: daily report email	108
23.3	Recipe 3: nightly file backup	109
23.4	Recipe 4: social post scheduler	110
24	Recipes: Business Workflows	111
24.1	Recipe 5: form to CRM	111
24.2	Recipe 6: lead enrichment	112
24.3	Recipe 7: invoice processing with AI	113
24.4	Recipe 8: Notion sync	114
24.5	Recipe 9: web page change monitor	114
25	Recipes: AI Workflows	116
25.1	Recipe 10: email triage with AI	116

25.2	Recipe 11: AI support chatbot with RAG	117
25.3	Recipe 12: Telegram assistant bot	118
26	Running Your Own n8n: Operations	120
26.1	Backups	121
26.2	Updating	122
26.3	Environment variables you'll actually use	123
26.4	Keeping the database small	124
26.5	Scaling with queue mode	124
26.6	Security hardening checklist	124
26.7	Command-line toolbox	125
27	What n8n Really Costs	126
27.1	What counts	126
27.2	n8n Cloud plans	127
27.3	Self-hosted costs	127
27.4	AI costs	128
27.5	Ten ways to use fewer executions	128
28	Templates and Community	129
28.1	Templates	129
28.2	Community nodes	130
28.3	Getting help	130
28.4	Keep learning	130
29	Troubleshooting	131
29.1	Triggers and publishing	131
29.2	Data and expressions	132
29.3	Credentials and APIs	132
29.4	AI	133
29.5	Self-hosting	133
30	Five Complete Projects	134
30.1	Project 1: Daily RSS digest	134
30.2	Project 2: Form to spreadsheet	135
30.3	Project 3: AI email summariser	135
30.4	Project 4: A tiny API	135
30.5	Project 5: Error alerts	136
31	Node and Expression Reference	137
31.1	Core nodes	137
31.2	Expression variables	138
31.3	Code node snippets	139
31.4	Cron cheat sheet	139
32	Troubleshooting and FAQ	141

33	Self-Hosting Environment Variable Reference	143
33.1	Setting variables	143
33.2	Deployment	144
33.3	Endpoints	147
33.4	Executions	149
33.5	Security	151
33.6	Database	152
33.7	Nodes	154
33.8	Logs	155
33.9	Timezone and Localization	157
33.10	Workflows	157
33.11	Credentials	157
33.12	Binary Data	158
33.13	Queue Mode	159
33.14	Task Runners	160
33.15	User Management and 2FA	163
34	Built-In Node Directory	165
	Quick Reference	174
	Glossary	175

CHAPTER 1

Meet n8n

Welcome. If you have ever copied information from an email into a spreadsheet, or checked the same website every morning, you already know the kind of work n8n is built to take off your hands. This chapter explains what n8n is, how it differs from similar tools, and what you will be able to build by the end of this manual.

IN THIS CHAPTER

- What workflow automation is, in plain English
- How n8n compares to Zapier and Make
- What “fair-code” and the Community Edition mean
- Cloud versus self-hosted, in one paragraph
- How this manual is organised and how to use it

1.1 What is workflow automation?

A **workflow** is a recipe that a computer follows automatically. It always starts with a **trigger** (*something happens*), then runs **steps** (*do these things*):

TRIGGER	STEPS
Every weekday at 8am	Get today's calendar → summarize with AI → send to Slack
A new row in Google Sheets	Create an invoice in Stripe → email the customer
A form is submitted	Add the person to your mailing list → notify you on Telegram
An email arrives at support@	AI sorts it → drafts a reply → creates a task

In n8n, every trigger and every step is a **node**: a box on a canvas. You connect the boxes with lines, and data flows along those lines from left to right.

Think of it like a kitchen. The trigger is the order coming in. Each node is a cook at a station: one chops, one fries, one plates. The lines between nodes are the counter where dishes slide from one cook to the next. You design the kitchen once; after that, every order is handled the same way, day and night.

1.2 What can you automate?

Almost anything that happens on a computer and follows a pattern. Some everyday examples that people build in their first month:

- A daily digest of news from your favourite websites
- Saving email attachments to Google Drive
- Posting new blog articles to social media
- Adding form sign-ups to a CRM
- Reminding customers about unpaid invoices
- Backing up files every night
- A chatbot that answers questions about your documents
- Alerts when a web page changes

If you can describe a task as “when *this* happens, do *that*, then *that*,” n8n can probably do it.

1.3 How n8n compares

	N8N	ZAPIER	MAKE
Style	Visual canvas with branches and loops	Simple step-by-step lists	Visual canvas
Self-hosting	Yes, free	No	No
Pricing basis	Per workflow <i>execution</i> , however many steps	Per task (each step counts)	Per operation
Code when you need it	JavaScript or Python nodes	Limited	Limited
AI agents and MCP	Built in	Yes	Yes
Best for	Power, flexibility, and low cost at scale	The quickest simple automations	Visual mid-complexity flows

The pricing row matters more than it looks. Because n8n charges per **execution** (one full run of a workflow) rather than per step, a twenty-step workflow costs the same as a two-step one. Complex automations stay cheap.

NOTE

n8n is “fair-code”: its source code is public and the self-hosted Community Edition is free to use, including for your own business, under n8n’s Sustainable Use License. You only pay for n8n Cloud or for enterprise features.

1.4 What “fair-code” means for you

You don’t need to read licences to use n8n, but it helps to know the basics:

- **You can** run the free self-hosted version for yourself or inside your own company, with unlimited workflows and executions.
- **You can** read and change the code.
- **You pay** if you want n8n to host it for you (n8n Cloud), or for business features such as single sign-on and Git version control on a self-hosted server.
- If you want to sell n8n itself as a hosted service to others, read the licence first.

1.5 Cloud or self-hosted?

- **n8n Cloud:** n8n runs it for you. Sign up and start in the browser. Easiest, with a free trial. **Recommended for beginners.**
- **Self-hosted:** you run n8n on your own computer or a server. Free and unlimited, but you’re responsible for keeping it running, updated, and secure.

You can start on Cloud and move later. Workflows export and import as files. Chapter 3 helps you decide properly.

1.6 How this manual is organised

The manual has five parts. Read the first part in order; after that, dip in wherever you need.

PART	CHAPTERS	WHAT YOU’LL LEARN
Getting started	1–7	Concepts, choosing and installing n8n, the editor, your first workflow
Building skills	8–18	Data, expressions, code, core nodes, triggers, credentials, reuse, errors, debugging
AI	19–22	AI steps, agents, knowledge bases (RAG), MCP
Recipes	23–25	Twelve complete, step-by-step workflows to copy
Running n8n	26–29	Self-hosting operations, costs, community, troubleshooting

 TIP

Keep n8n open in another window while you read, and try each example as you go. Automation is learned by doing: ten minutes of clicking teaches more than an hour of reading.

- ☐ I know what a workflow, trigger and node are
- ☐ I know n8n charges per execution, not per step
- ☐ I know there is a free self-hosted edition and a paid Cloud service
- ☐ I have n8n open (or will after Chapter 4) so I can follow along

CHAPTER 2

Key Ideas in Five Minutes

Before you click anything, learn a handful of words. Every n8n screen, error message and forum answer uses them, and once they click, everything else in this manual gets easier.

IN THIS CHAPTER

- The ten words you need to know
- Items: the single most important idea
- Draft versus published workflows
- Manual, partial and production runs

TERM	MEANING
Workflow	A set of connected nodes that does one job
Node	One block: a trigger, an action in an app, or a logic step
Trigger node	The first node, which starts the workflow (a schedule, a webhook, a new email...)
Item	One piece of data moving through the workflow, like one email or one spreadsheet row. Nodes run once per item
JSON	The text format n8n uses to show data: <code>{ "name": "Sam", "city": "Austin" }</code>
Expression	A small formula inside <code>{{ }}</code> that pulls in data from earlier nodes
Credential	Saved sign-in details that let a node use an app on your behalf
Execution	One run of a workflow; n8n keeps a log of each
Publish	Make a workflow live, so its triggers fire automatically
Instance	Your own copy of n8n, on Cloud or on your server, with its own address

2.1 Items: the most important idea

When a node outputs **five rows** from a spreadsheet, that's **five items**, and the next node normally runs **five times**, once for each. That's how a “send an email” node can send one personalized email per row without any loops.

Picture a conveyor belt. Each item is a parcel on the belt. Each node is a machine the belt passes through. The machine does its job to every parcel in turn, then passes them all along. You rarely have to think about “for each row, do this”: n8n does it for you.

SEE ITEMS IN ACTION

1 Imagine a sheet with three customers.

A Google Sheets node reads it and outputs three items: Ana, Ben and Chloé.

2 Add a Gmail node after it.

Set **To** to the expression `{{ $json.email }}`.

3 Run it.

The Gmail node runs three times and sends three emails, each to the right person, because `$json` always means “the current item.”

 TIP

When something sends one email too many, or one too few, count the items. Nine times out of ten the surprise is that a node produced more (or fewer) items than you expected. Chapter 8 shows how to inspect them.

2.2 Nodes come in a few families

FAMILY	WHAT IT DOES	EXAMPLES
Trigger	Starts the workflow	Schedule, Webhook, Gmail Trigger
App (action)	Does something in another service	Slack, Google Sheets, Notion
Core	Logic and data shaping	If, Merge, Edit Fields, Code
AI	Language models and agents	AI Agent, chat models, vector stores

Some AI nodes are **cluster nodes**: a main **root node** (such as AI Agent) with smaller **sub-nodes** plugged into its underside (a chat model, a memory, some tools). You'll meet them in Chapter 19.

2.3 Draft versus published

n8n saves your edits automatically as a **draft**. Your triggers only run automatically after you click **Publish** (or press `Shift` + `P`). Later edits stay in draft until you publish again, so you can safely tinker with a live workflow.

2.4 Three kinds of run

KIND	STARTED BY	COUNTS AGAINST CLOUD QUOTA?
Manual	You, clicking Execute workflow or Execute step	No
Partial	You, running one node and the ones it depends on	No
Production	A trigger on a published workflow	Yes

Only production runs count as executions on n8n Cloud plans, so test as much as you like.

- ☐ I can explain the difference between a node and an item
- ☐ I know `$json` means “the current item”
- ☐ I know edits stay in draft until I publish
- ☐ I know manual test runs don’t use up my execution allowance

Cloud or Self-Hosted? Making the Choice

There are two ways to get n8n, and both run the same product. The question is simply who looks after the computer it runs on: n8n, or you. This chapter lays out the trade-offs so you can choose with confidence, and reassures you that the choice isn't permanent.

IN THIS CHAPTER

- What each option means in practice
- A side-by-side comparison
- A quick decision guide
- The free self-hosted editions
- How to switch later

3.1 The two options in plain English

n8n Cloud is like renting a fully serviced flat. You sign up, get an address such as `yourname.app.n8n.cloud`, and start building in your browser. n8n handles the server, security patches, updates, backups of the platform and scaling. You pay a monthly fee based on how many workflow runs you need.

Self-hosted is like owning a house. You install n8n on your own computer or on a rented server in a data centre (a **VPS**, or *virtual private server*). The software is free and has no execution limits, but you fix the roof: installation, updates, backups, security and keeping it switched on are your job.

3.2 Side by side

	N8N CLOUD	SELF-HOSTED
Setup time	Minutes	An hour or more
Technical skill	None	Comfortable with a terminal
Software cost	From €20/month (yearly)	Free Community Edition
Server cost	Included	A small VPS, a few euros a month
Execution limits	By plan	None (only your server's power)
Updates and security	Done for you	Your job
Your data's location	n8n's servers	Wherever you choose
Custom code libraries	Limited	Allowed
AI without API keys	Gateway credits	Bring your own keys

3.3 Choose n8n Cloud if...

- you are new to n8n, servers or the command line;
- you want to be building within five minutes;
- you'd rather spend your time on workflows than on maintenance;
- you need webhooks reachable from the internet straight away, with HTTPS already set up.

3.4 Choose self-hosting if...

- you are comfortable in a terminal, or keen to learn;
- you want unlimited executions for a fixed, low server cost;
- your data must stay on your own infrastructure (for example for privacy rules);
- you need custom npm or Python packages in Code nodes, your own custom nodes, or bash commands.

☆ IMPORTANT

n8n itself recommends self-hosting for people with server experience. Mistakes can lead to data loss, security problems or downtime. If that sounds daunting, start on Cloud. Nothing you learn is wasted: the editor is identical.

3.5 A quick decision guide

DECIDE IN FOUR QUESTIONS

1 Do you just want to learn?

Start a Cloud free trial, or run n8n on your laptop with Docker (Chapter 4). Both are fine for practice.

2 Will it run real work for you or your business?

Then it must be always on. Choose Cloud, or a VPS with Docker Compose (Chapter 5).

3 Will you run thousands of executions a month?

Self-hosting is usually far cheaper at high volume.

4 Does anyone need to approve where data lives?

If yes, self-host somewhere they approve, or ask n8n about Enterprise options.

3.6 The self-hosted editions

Self-hosted n8n comes in four flavours. All run the same software; a licence key unlocks extra features.

EDITION	COST	WHAT YOU GET
Community	Free	Almost the complete feature set
Registered Community	Free	Adds folders, debug in editor and custom execution data
Business	Paid	SSO, environments, projects, Git version control
Enterprise	Paid	Advanced security, scaling and governance

To register a Community instance for the free extras, open **Settings** > **Usage and plan** and select **Unlock**. You'll receive a licence key by email.

NOTE

Features that need a paid plan include custom variables, environments, external secrets, log streaming, multi-main mode, projects, SSO, workflow and credential sharing, and Git version control. Feature lists change, so check n8n.io/pricing for the current details.

3.7 Switching later

You aren't locked in. Workflows are stored as JSON files that you can download from one instance and import into another (Chapter 16). Credentials (your passwords and tokens) don't travel with downloaded workflows, so you'd reconnect your apps after a move. That takes minutes per app.

- ☐ I know who maintains the server in each option
- ☐ I have picked Cloud or self-hosted for now
- ☐ I know I can move later by exporting workflows
- ☐ If self-hosting for real, I'm ready to read Chapters 5 and 26

Get n8n Running

Time to get your own n8n. This chapter covers the two quickest routes: an n8n Cloud trial in your browser, or n8n on your own computer with Docker. If you want an always-on server with your own domain name, read this chapter first and then Chapter 5.

IN THIS CHAPTER

- Starting an n8n Cloud trial, and what happens when it ends
- The Cloud plans and their limits
- Running n8n on your computer: the one-line setup, Docker, and npx
- Creating the owner account and your first look around

4.1 Option A: n8n Cloud (recommended)

START ON N8N CLOUD

- 1 Go to n8n.io** and click **Get started**. Create an account; a free trial starts automatically. No credit card is needed.
- 2 Choose a name** for your workspace. It becomes your address, like `yourname.app.n8n.cloud`.
- 3 Open the editor.**
You'll see the Overview page with your workflows list.
- 4 Check your time zone.**
Schedules use it. On Cloud, n8n tries to detect your time zone at sign-up. To change it, open your Cloud dashboard, select **Manage**, and change the **Timezone** dropdown.

What the trial includes

The free trial lasts **14 days** and gives you Pro plan features with the computing power of the Starter plan. Its limits are **1,000 executions**, 5 concurrent executions and one team project. You can upgrade at any time with the **Upgrade** button in the top-right corner, choosing yearly or monthly billing.

⚠ WHEN THE TRIAL ENDS

If you don't upgrade, the trial expires on its own and no charges occur, but n8n then deletes your workspace. You have 90 days to download your workflows afterwards. Export anything you want to keep before day 14.

The Cloud plans

CLOUD PLAN	PRICE (BILLED YEARLY)	EXECUTIONS PER MONTH	HIGHLIGHTS
Starter	€20/month	2,500	Unlimited workflows, AI agents and MCP, 5 concurrent runs
Pro	€50/month	10,000	More concurrency, admin roles, workflow history, insights
Enterprise	Contact sales	Custom	SSO, advanced security, support with an SLA

Prices from n8n.io/pricing, October 2026. Billed monthly, Starter is €24 and Pro is €60 a month. Every plan includes unlimited users, unlimited workflows and every integration. An **execution** is one full run of a workflow, however many nodes it has. Chapter 27 covers costs in depth, including the self-hosted Business plan.

4.2 Option B: self-host on your own computer

Running n8n on your laptop is free and great for learning. You need **Docker**, a free program that runs software in sealed boxes called **containers**, so n8n can't interfere with the rest of your computer. Install **Docker Desktop** from docker.com, open it once, and leave it running in the background.

The one-line setup (easiest)

On macOS, Linux, or Windows with WSL (Windows Subsystem for Linux), open a terminal and run:

TERMINAL

```
curl -fsSL https://get.n8n.io | sh
```

The script checks Docker is running, creates an `n8n` folder in the current directory, writes the configuration files, downloads n8n and starts it. When it finishes it prints `n8n is running at: http://localhost:5678`.

TO DO THIS	RUN THIS
Stop n8n	<code>docker compose -f ./n8n/compose.yml down</code>
Start it again	<code>docker compose -f ./n8n/compose.yml up -d</code>
Upgrade	<code>curl -fsSL https://get.n8n.io sh -s -- --upgrade</code>

TIP

Prefer not to pipe a script from the internet straight into your shell? Download it with `curl -fsSL https://get.n8n.io -o get-n8n.sh`, read it with `less get-n8n.sh`, then run `sh get-n8n.sh`.

DANGER

The uninstall command, `docker compose -f ./n8n/compose.yml down -v`, deletes **all** your n8n data, including workflows and credentials. The `-v` means “remove volumes.” Leave it off when you only want to stop n8n.

With plain Docker

If you’d rather type the Docker command yourself (any system with Docker Desktop installed):

TERMINAL

```
docker volume create n8n_data

docker run -it --rm --name n8n -p 5678:5678 \
  -e GENERIC_TIMEZONE="America/New_York" -e TZ="America/New_York" \
  -v n8n_data:/home/node/.n8n \
  n8nio/n8n
```

What each part means:

PART	MEANING
<code>docker volume create n8n_data</code>	Make a storage box that survives restarts
<code>-p 5678:5678</code>	Open n8n on port 5678 of your computer
<code>-e GENERIC_TIMEZONE=...</code>	Set the time zone schedules use
<code>-v n8n_data:/home/node/.n8n</code>	Keep n8n’s data in that storage box
<code>--rm</code>	Delete the container (not the data) when it stops

Just trying it? If you have Node.js (version 20.19–24) installed, run `npm install n8n`.

Then open <http://localhost:5678> in your browser and create the owner account.

 NOTE

The npm installation method is deprecated from n8n 3.0, when n8n will be distributed only through Docker. Use Docker or Docker Compose for anything long-lived.

 WARNING

A self-hosted n8n on your laptop only runs while your laptop is on and awake, and outside services can't reach its webhooks. For always-on automations, use n8n Cloud or a small cloud server (see Chapter 5).

 TIP

Replace `America/New_York` with your own time zone (for example `Europe/London` or `America/Los_Angeles`) so schedules fire at the right local time. Self-hosted n8n uses New York time if you don't set one.

4.3 Create the owner account

The first person to open a new self-hosted instance becomes its **owner**: the administrator who can invite other users and change settings.

SET UP YOUR INSTANCE

- 1 Open <http://localhost:5678>.**
You'll see the **Set up owner account** screen.
- 2 Enter your email, name and a strong password.**
Store the password in a password manager.
- 3 Answer the short survey** (or skip it). It only tailors suggestions.
- 4 Register for free extras (optional).**
Open **Settings** > **Usage and plan** and select **Unlock**. Registering your email unlocks folders, debug in editor and custom execution data, at no cost.

4.4 Turn on two-factor authentication

Two-factor authentication (2FA) means signing in needs your password *and* a code from an app on your phone. Do this on Cloud and self-hosted alike: your n8n holds the keys to all your other apps.

ENABLE 2FA

- 1 Open your personal settings.**
Go to `Settings` > `Personal`.
- 2 Select Enable 2FA.**
n8n shows a QR code. Scan it with an authenticator app such as Google Authenticator, 1Password or Authy.
- 3 Enter the six-digit code** from the app in **Code from authenticator app**, then select **Continue**.
- 4 Save the recovery codes** somewhere safe, away from your computer. They get you back in if you lose your phone.

- ☐ I have a Cloud trial or a local n8n running
- ☐ I can open the editor in my browser
- ☐ My time zone is set correctly
- ☐ I've created the owner account and turned on 2FA
- ☐ I know `down -v` deletes my data, and plain `down` doesn't

Self-Hosting on a Server

A laptop is fine for learning, but real automations need a computer that never sleeps and that the internet can reach. This chapter walks you through renting a small cloud server and running n8n on it with Docker Compose, a proper PostgreSQL database, your own domain name and automatic HTTPS. Take it slowly: every command is explained.

IN THIS CHAPTER

- What a VPS, a domain, DNS, HTTPS and a reverse proxy are
- Renting and securing a small Linux server
- A complete Docker Compose setup: n8n, PostgreSQL and Caddy
- The environment variables that make webhooks work
- Other routes: the n8n Caddy repository, Traefik, and npm

☆ IMPORTANT

Self-hosting means you are the system administrator. You'll be responsible for updates, backups and security (Chapter 26). If any of this feels like too much, n8n Cloud does all of it for you.

5.1 The pieces, explained

PIECE	WHAT IT IS	ANALOGY
VPS	A rented computer in a data centre, always on	A flat you rent
Domain	A name like <code>example.com</code> you buy from a registrar	Your street address
DNS record	A setting that points <code>n8n.example.com</code> at your server's IP address	The signpost to your door
HTTPS	Encrypted web traffic, shown by the padlock in the browser	Sealed envelopes instead of postcards
Reverse proxy	A program that receives web traffic and hands it to n8n, handling HTTPS certificates	A receptionist at the front desk
Docker Compose	A tool that starts several containers together from one file	A recipe card for the whole kitchen
PostgreSQL	A robust database for n8n's workflows and history	A filing cabinet built for heavy use

We'll use **Caddy** as the reverse proxy because it fetches and renews free HTTPS certificates from Let's Encrypt on its own, with a three-line configuration file.

5.2 Step 1: rent a server

Any provider works: Hetzner, DigitalOcean, AWS, Azure, Google Cloud and many others. n8n publishes guides for several of them. Choose:

- **Operating system:** Ubuntu LTS (the long-term-support version) is the most common choice and matches the commands below.
- **Size:** n8n's own Hetzner guide says the small CPX11 type is enough for most usage. Choose at least **4 GB RAM and 2 vCPUs** if you plan to turn on n8n Assistant, which runs a code sandbox.
- **Login:** choose an **SSH key** rather than a password. SSH is how you'll log in to type commands; keys are far safer.

When the server is ready, note its **IP address** (four numbers like `203.0.113.10`) and log in from your terminal:

TERMINAL

```
ssh root@203.0.113.10
```

5.3 Step 2: point your domain at it

In your domain registrar's DNS settings, add an **A record**:

RECORD TYPE	NAME	DESTINATION
A	n8n	Your server's IP address

This makes `n8n.example.com` lead to your server. DNS changes can take from a few minutes to a few hours to spread. Check with `ping n8n.example.com` from your own computer: when it shows your server's IP, you're ready.

5.4 Step 3: install Docker and open the firewall

On the server, install Docker with Docker's official convenience script, then check it worked:

TERMINAL

```
curl -fsSL https://get.docker.com | sh
docker --version
docker compose version
```

Now set up the firewall. **ufw** (“uncomplicated firewall”) comes with Ubuntu. Allow SSH first so you don't lock yourself out, then web traffic:

TERMINAL

```
sudo ufw allow OpenSSH
sudo ufw allow 80
sudo ufw allow 443
sudo ufw enable
```

Port 80 is plain web traffic (Caddy needs it to prove you own the domain and to redirect visitors to HTTPS). Port 443 is HTTPS. You do **not** open port 5678: only Caddy talks to n8n directly.

5.5 Step 4: create the project folder and secrets

TERMINAL

```
mkdir ~/n8n-compose && cd ~/n8n-compose
openssl rand -hex 32
```

The second command prints a long random string. Copy it: it becomes your **encryption key**, which n8n uses to scramble your saved credentials. Run `openssl rand -hex 32` again for a database password. Now create a file called `.env` with `nano .env` and paste this, replacing

every value:

.ENV

```
N8N_VERSION=2.43.6
N8N_DOMAIN=n8n.example.com
GENERIC_TIMEZONE=Europe/London

N8N_ENCRYPTION_KEY=paste-your-first-random-string-here

POSTGRES_USER=n8n_admin
POSTGRES_PASSWORD=paste-your-second-random-string-here
POSTGRES_DB=n8n
```

Save with **Ctrl** + **O**, **Enter**, then exit with **Ctrl** + **X**.

 TIP

`N8N_VERSION` pins the exact n8n version, so an update only happens when you choose. Find the latest stable version on n8n's GitHub releases page (github.com/n8n-io/n8n/releases). At the time of writing the stable release was 2.43.6.

 DANGER

Store a copy of `.env`, especially `N8N_ENCRYPTION_KEY`, in your password manager. If you lose the key, every saved credential in n8n becomes unreadable and must be re-entered.

5.6 Step 5: tell Caddy about your domain

Create a file named `Caddyfile` (no extension) with `nano Caddyfile`:

CADDYFILE

```
n8n.example.com {
  reverse_proxy n8n:5678 {
    flush_interval -1
  }
}
```

Replace `n8n.example.com` with your domain. `reverse_proxy n8n:5678` sends visitors to the container named `n8n` on port 5678. `flush_interval -1` passes streamed data (like live AI chat replies) straight through instead of buffering it.

5.7 Step 6: write the Compose file

Create `compose.yml` with `nano compose.yml`. The file is long, so it's shown in two parts. Paste **part 1** first, then **part 2** directly underneath it in the same file. Indentation (the spaces at the start of lines) matters in YAML, so keep it exactly as shown.

Part 1 defines Caddy and the PostgreSQL database:

COMPOSE.YML (PART 1)

```
services:
  caddy:
    image: caddy:latest
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - caddy_data:/data
      - ./Caddyfile:/etc/caddy/Caddyfile

  postgres:
    image: postgres:18
    restart: always
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
      PGDATA: /var/lib/postgresql/data
    volumes:
      - db_storage:/var/lib/postgresql/data
    healthcheck:
      test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER} -d ${POSTGRES_DB}"]
      interval: 5s
      timeout: 5s
      retries: 10
```

The `${...}` placeholders are filled in from your `.env` file. The **healthcheck** lets Docker know when the database is ready, so n8n doesn't start too early. Keep the `PGDATA` line: PostgreSQL 18 changed its default data folder, and this line keeps the data where the volume expects it.

Part 2 defines n8n itself and the storage volumes:

COMPOSE.YML (PART 2)

```

n8n:
  image: docker.n8n.io/n8nio/n8n:${N8N_VERSION}
  restart: always
  environment:
    DB_TYPE: postgresdb
    DB_POSTGRESDB_HOST: postgres
    DB_POSTGRESDB_PORT: "5432"
    DB_POSTGRESDB_DATABASE: ${POSTGRES_DB}
    DB_POSTGRESDB_USER: ${POSTGRES_USER}
    DB_POSTGRESDB_PASSWORD: ${POSTGRES_PASSWORD}
    N8N_HOST: ${N8N_DOMAIN}
    N8N_PORT: "5678"
    N8N_PROTOCOL: https
    N8N_WEBHOOK_URL: https://${N8N_DOMAIN}/
    N8N_PROXY_HOPS: "1"
    N8N_ENCRYPTION_KEY: ${N8N_ENCRYPTION_KEY}
    NODE_ENV: production
    GENERIC_TIMEZONE: ${GENERIC_TIMEZONE}
    TZ: ${GENERIC_TIMEZONE}
  volumes:
    - n8n_data:/home/node/.n8n
  depends_on:
    postgres:
      condition: service_healthy

volumes:
  caddy_data:
  db_storage:
  n8n_data:

```

5.8 What those settings do

SETTING	WHY IT MATTERS
<code>DB_TYPE</code> and <code>DB_POSTGRESDB_*</code>	Use PostgreSQL instead of the built-in SQLite file
<code>N8N_HOST</code> , <code>N8N_PROTOCOL</code>	Tell n8n its public name and that it's served over HTTPS
<code>N8N_WEBHOOK_URL</code>	The public address n8n shows and registers for webhooks
<code>N8N_PROXY_HOPS</code>	n8n sits behind one proxy (Caddy), so trust its forwarded headers
<code>N8N_ENCRYPTION_KEY</code>	Your own key for encrypting credentials
<code>TZ</code> , <code>GENERIC_TIMEZONE</code>	The time zone for schedules and dates
<code>restart: always</code>	Start again automatically after a crash or reboot

NOTE

`N8N_WEBHOOK_URL` is available from n8n 2.30.0. The older name, `WEBHOOK_URL`, still works but is deprecated from 2.35.0, so you'll see it in older tutorials.

5.9 Step 7: start everything

TERMINAL

```
docker compose up -d
docker compose ps
docker compose logs -f n8n
```

`up -d` starts all three containers in the background (“detached”). `ps` lists them; all should say `running` (Postgres also `healthy`). `logs -f` follows n8n’s log; press **Ctrl** + **C** to stop watching (n8n keeps running). The first start takes a minute or two while Caddy fetches your certificate.

Open `https://n8n.example.com`. You should see the padlock and the owner setup screen. Create the owner account straight away, before anyone else finds the page, and turn on 2FA (Chapter 4).

5.10 Everyday commands

TO DO THIS	RUN THIS (IN <code>~/N8N-COMPOSE</code>)
See what’s running	<code>docker compose ps</code>
Read n8n’s recent log	<code>docker compose logs --tail=100 n8n</code>
Restart n8n	<code>docker compose restart n8n</code>
Stop everything	<code>docker compose down</code>
Start everything	<code>docker compose up -d</code>
Apply <code>.env</code> changes	<code>docker compose up -d</code>

5.11 Other ways to self-host

- **n8n’s Caddy repository.** n8n’s Hetzner and DigitalOcean guides clone `github.com/n8n-io/n8n-docker-caddy`, a ready-made Caddy setup using SQLite. It’s a good alternative if you’d rather not write files by hand.
- **Traefik.** n8n’s Docker Compose server guide uses Traefik, another reverse proxy that obtains certificates automatically, configured with labels in the Compose file.

- **The n8n-hosting repository** (`github.com/n8n-io/n8n-hosting`) has more Compose examples, including a hardened PostgreSQL setup with a separate task runner container.
- **npm.** `npm install n8n -g` then `n8n start` still works with Node.js 20.19–24, but npm installs are deprecated from n8n 3.0 and don't support n8n Assistant. Prefer Docker.
- **Testing webhooks locally.** n8n's docs describe a development tunnel using cloudflared. It's for local testing only, never production.

5.12 If it doesn't work

SYMPTOM	LIKELY CAUSE	FIX
Browser can't reach the site	DNS not updated yet, or firewall	Wait; check <code>ping</code> ; check <code>sudo ufw status</code>
Certificate error	Port 80 blocked, or DNS points elsewhere	Open port 80; fix the A record; check <code>docker compose logs caddy</code>
n8n keeps restarting	Database not ready or wrong password	Check <code>docker compose logs n8n</code> ; compare <code>.env</code> values
Webhook URLs show <code>localhost</code>	<code>N8N_WEBHOOK_URL</code> missing	Add it, then <code>docker compose up -d</code>
"Credentials could not be decrypted"	Encryption key changed	Restore the original <code>N8N_ENCRYPTION_KEY</code>

- ☐ My server has a domain name pointing at it
- ☐ The firewall allows only SSH, 80 and 443
- ☐ `.env` holds my secrets, and a copy is in my password manager
- ☐ n8n opens over HTTPS with a padlock
- ☐ I created the owner account and enabled 2FA immediately
- ☐ I've read Chapter 26 on backups and updates

A Tour of the Editor

The n8n editor is where you'll spend most of your time. It looks busy at first, but there are only a few areas, and each has one job. This chapter walks you around them, shows the shortcuts that save the most time, and explains the three-panel window you'll use to set up every node.

IN THIS CHAPTER

- The Overview page, projects and folders
- The canvas, the nodes panel and the top bar
- The node details view: input, parameters, output
- Sticky notes, groups, and the command bar
- Keyboard shortcuts worth learning

6.1 The Overview page

When you sign in, you land on **Overview**: a list of everything you can access. Tabs along the top switch between **Workflows**, **Credentials**, **Executions** and **Data tables**.

- **Personal space** holds the workflows and credentials you created.
- **Projects** group workflows, credentials and data tables for a team. The number of shared projects depends on your plan.
- **Folders** let you organise workflows inside a project. On self-hosted Community, folders need the free registration.
- **Tags** are labels you add to workflows, like `marketing` or `client-acme`, then filter by.

6.2 The main areas

- **Left sidebar:** Overview (all workflows), Templates, Credentials, and Settings.
- **Canvas:** the big grid where you place and connect nodes. Scroll to zoom; drag the background to pan.
- **Top bar:** the workflow's name, the **Publish** button, and the **Editor / Executions** tabs.
- **Execute workflow** button (bottom of canvas): runs the whole workflow once, right now, for testing.

6.3 Adding nodes

Click the + on the canvas (or press **N**) to open the **nodes panel**. Search by app ("Gmail") or by action ("send email"). Nodes are grouped as:

- **Triggers:** how a workflow starts.
- **App nodes:** actions in other services (Google Sheets, Slack, Notion, Airtable...).
- **Core nodes:** logic and data: If, Switch, Edit Fields (Set), Merge, Loop Over Items, Wait, Code, HTTP Request.
- **AI nodes:** AI Agent, chat models, memory, tools, vector stores.

Many app nodes contain dozens of **operations**. The Google Sheets node, for example, can append rows, update rows, read rows, delete rows and more. Pick the node, then choose the operation.

6.4 Connecting nodes

Every node has small circles called **connectors**. Drag from the output connector on the right side of one node to the input connector on the left of another to join them. To insert a node between two others, hover over the line joining them and click the + that appears. To delete a connection, hover over it and click the bin icon.

6.5 The node details view

Double-click any node to open it. You'll see three panels:

PANEL	WHAT IT SHOWS
Input (left)	The data arriving from the previous node
Parameters (middle)	The node's settings: which account, which operation, which fields
Output (right)	What the node produced after you click Execute step

Switch the input and output views between **Table**, **JSON**, and **Schema**. Table is easiest to read; JSON shows the exact structure.

The middle panel has two tabs. **Parameters** holds what the node does. **Settings** holds how it behaves: retry on failure, what to do on error, whether to always output data, and notes to show on the canvas.

TIP

Test every node as you add it. Click **Execute step**, check the output looks right, then move on. Building in small tested steps is the secret to workflows that work first time.

6.6 Fixed values and expressions

Every parameter can be **Fixed** (a plain value you type) or an **Expression** (a formula that uses data from earlier nodes). Hover over a field to see the **Fixed / Expression** switch. A shortcut: type `=` in an empty field to switch it to expression mode. Chapter 9 covers expressions in depth.

6.7 Sticky notes and groups

Sticky notes are coloured notes you place on the canvas to explain what a section does. Press `Shift + S` to add one. They support Markdown, so you can add headings, lists and links.

Groups box related nodes together: select nodes and press `Ctrl / Cmd + G`.

TIP

Future-you is a different person. A one-line sticky note (“Runs at 7am, emails Sam the weather”) saves twenty minutes of puzzling six months from now.

6.8 The command bar

Press `Ctrl / Cmd + K` to open the **command bar**. Type what you want: add a node, publish, open a workflow, open credentials, jump to settings, or retry an execution. It’s the fastest way around n8n once you know it exists.

6.9 Keyboard shortcuts worth learning

SHORTCUT	DOES
<code>N</code>	Open the nodes panel
<code>Ctrl / Cmd + Enter</code>	Execute the workflow
<code>Shift + P</code>	Publish

Ctrl / Cmd + Z	Undo (add Shift to redo)
Ctrl / Cmd + K	Command bar
1	Zoom to fit the whole workflow
0	Reset zoom
Space + drag	Move around the canvas
Shift + S	Add a sticky note
F2	Rename the selected node
D	Deactivate (skip) the selected node
P	Pin the selected node's data
Delete	Delete the selected node
Ctrl / Cmd + C / V	Copy and paste nodes

NOTE

Copied nodes are plain JSON text. You can paste them into a message to a colleague or a forum post, and they can paste them straight onto their canvas.

6.10 The Executions tab

Next to **Editor** at the top of each workflow is **Executions**: a list of every run of that workflow, with its status and timing. Click one to see exactly what each node received and produced. Chapter 18 shows how to use it for debugging.

6.11 Settings you'll visit

PLACE	WHAT'S THERE
Settings > Personal	Your name, password and 2FA
Settings > Usage and plan	Plan, licence, free registration
Settings > Community nodes	Install extra nodes (self-hosted)
Settings > Instance-level MCP	Let AI apps use your n8n (Chapter 22)
Workflow menu (...) > Settings	Error workflow, time zone, what executions to save

- ☐ I can open the nodes panel and add a node
- ☐ I can connect, insert and delete connections
- ☐ I know the three panels of the node details view

Your First Workflow: A Morning Weather Email

Let's build a workflow that checks the weather every morning and emails you a summary. It uses a free weather service, Open-Meteo, that needs no account. Along the way you'll use a trigger, an HTTP request, an expression and an app node: the four skills behind almost every automation.

IN THIS CHAPTER

- Build a complete scheduled workflow, step by step
- Read data from a web service with HTTP Request
- Write your first expression
- Add a decision with the If node
- Publish it and check that it ran

BUILD THE WEATHER WORKFLOW

1

Create a workflow.

On the Overview page click **Create Workflow**. Name it "Morning weather".

2

Add a Schedule Trigger.

Click **Add first step**, search **Schedule**, and choose **Schedule Trigger**. Set **Trigger Interval** to **Days** and **Trigger at Hour** to **7am**. Close the node.

3 Add an HTTP Request node.

Click the **+** after the trigger, search **HTTP Request**, and paste this into **URL** (it's New York; change the latitude and longitude to your city):

TEXT

```
https://api.open-meteo.com/v1/forecast?
latitude=40.71&longitude=-74.01&daily=temperature_2m_max,temperature_2m_min,precipitation_proba
bility_max&temperature_unit=fahrenheit&timezone=auto
```

Click **Execute step**. The output shows today's forecast in a `daily` section.

4 Add an Edit Fields (Set) node to make a friendly message. Add a field named `message` and switch its value to **Expression mode, then enter:**

TEXT

```
Today: high {{ $json.daily.temperature_2m_max[0] }}°F, low {{ $json.daily.temperature_2m_min[0]
}}°F, {{ $json.daily.precipitation_probability_max[0] }}% chance of rain.
```

Click **Execute step** and check the message reads correctly.

5 Add a Gmail node (or Outlook, or Send Email). Choose the **Send a message operation. Under **Credential**, choose **Create new credential** and sign in with Google. Set **To** to your address, **Subject** to `Your morning weather`, and drag the `message` field from the input panel into **Message**.****6 Test the whole thing.**

Click **Execute workflow** and check your inbox.

7 Publish.

Click **Publish** (top right). From tomorrow, it runs every morning at 7.

Congratulations: you've built a real automation. Notice the pattern you'll use forever: **trigger** → **get data** → **reshape data** → **send it somewhere**.

NOTE

The `[0]` in the expression means “the first entry in the list,” which is today. `[1]` would be tomorrow.

7.1 Finding your latitude and longitude

Open Google Maps or OpenStreetMap, right-click your town, and the coordinates appear (two numbers such as `51.51, -0.13` for London). The first is latitude, the second longitude. Prefer Celsius? Remove `&temperature_unit=fahrenheit` from the URL and change `°F` to `°C` in your message.

7.2 What just happened, node by node

NODE	INPUT	OUTPUT
Schedule Trigger	Nothing: it starts the run	One item with the time it fired
HTTP Request	That item	One item: the forecast as JSON
Edit Fields	The forecast	One item with a <code>message</code> field
Gmail	The message	One item describing the sent email

Each node received exactly one item, so each ran once. Open each node after an execution to see these panels for yourself.

7.3 Add a decision

Make it smarter: only send an umbrella warning if rain is likely.

ADD AN IF NODE

- 1 Insert an If node** between Edit Fields and Gmail. Hover over the connecting line and click **+**.
- 2 Set the condition.**
Value 1: drag in `precipitation_probability_max` (or use the expression `{{ $('HTTP Request').item.json.daily.precipitation_probability_max[0] }}`). Operation: **Number > is greater than**. Value 2: `50`.
- 3 Connect the outputs.**
Connect the **true** output to your Gmail node with the subject “☔ Take an umbrella!”, and add a second Gmail node on **false** for normal days, or leave false empty to send nothing.

The expression in step 2 uses `$('HTTP Request')`, which means “the output of the node called HTTP Request.” That’s how you reach back past the previous node to any earlier one. If you rename that node, n8n updates expressions that refer to it.

7.4 Check that it really ran

The morning after you publish, open the workflow and click the **Executions** tab. You should see one run at 7:00 with a green **Succeeded** status. Click it to replay exactly what happened. If it's red, click the failed node to read the error, and see Chapter 18.

TIP

Don't want to wait until tomorrow? Temporarily set the schedule to **Minutes** with **Minutes Between Triggers** set to `2`, publish, watch two runs appear in Executions, then set it back to 7am and publish again.

7.5 Make it yours

Try one of these changes on your own. Each takes under five minutes:

- Send to Slack or Telegram instead of email
 - Add tomorrow's forecast with `[1]`
 - Run only on weekdays (Weeks interval)
 - Add the city name to the subject
 - Send a different message if it will be hot
 - Log each forecast to Google Sheets
-
- ☐ My workflow ran successfully with **Execute workflow**
 - ☐ I understand what `$json` and `[0]` mean
 - ☐ I added an If node with true and false paths
 - ☐ I published it and saw a production run in Executions

How Data Flows: Items, JSON and Binary

Every confusing moment in n8n comes down to one question: *what data does this node actually have?* This chapter gives you a clear picture of how n8n packages data, how it moves between nodes, how files travel alongside it, and how to inspect it at every step.

IN THIS CHAPTER

- JSON in five minutes
- The shape of n8n data: a list of items, each with `json` and `binary`
- How nodes process many items
- Files (binary data) and the nodes that handle them
- Splitting, combining and linking items
- Pinning and mocking data while you build

8.1 JSON in five minutes

JSON (JavaScript Object Notation) is a way of writing structured data as text. It has just a few building blocks:

BUILDING BLOCK	LOOKS LIKE	HOLDS
String	"Sam"	Text, always in double quotes
Number	42 or 3.5	Numbers, no quotes
Boolean	true / false	Yes or no
Null	null	"Nothing here"
Object	{ "name": "Sam" }	Named fields (keys and values)
Array	["a", "b", "c"]	An ordered list, counted from 0

Objects and arrays can sit inside each other as deeply as you like:

```
JSON
{
  "name": "Sam Lee",
  "email": "sam@example.com",
  "address": { "city": "Austin", "zip": "78701" },
  "orders": [ 120, 45, 300 ]
}
```

| **Schedule Trigger** | A time arrives: every hour, every Monday at 9... | Weekly report | | **Webhook** | Another app sends data to n8n's unique URL | Payment received in Stripe | | **n8n Form Trigger** | Someone submits a form n8n hosts for you | Event sign-up | | **App triggers** | Something happens in an app (new email, new row, new file) | New Gmail message with a label | | **Chat Trigger** | Someone sends a message in a chat window | A chatbot for your website | | **Execute Sub-workflow Trigger** | Another workflow calls this one | Reusable "send notification" sub-workflow | | **Error Trigger** | Another workflow fails | Alert yourself on failure | | **Manual** | You click **Execute workflow** | Testing |

To reach a value you follow the path: `address.city` is "Austin", and `orders[0]` is 120, the first entry.

8.2 The shape of n8n data

All data passed between nodes is a **list of items**. Each item is an object with a `json` key holding the actual data, and optionally a `binary` key holding files:

```
JSON
[
  { "json": { "name": "Ana", "email": "ana@example.com" } },
  { "json": { "name": "Ben", "email": "ben@example.com" } }
]
```

That's two items. In the editor's Table view you'd see two rows. In JSON view n8n usually hides the `json` wrapper to keep things tidy, but it's always there underneath.

TIP

In the Code node you can return plain objects, and n8n adds the `json` wrapper and the surrounding list for you. When something says “item,” think “one row” or “one record.”

8.3 How nodes process many items

Most nodes run their operation **once per item**. If a Google Sheets node outputs 50 rows and you connect a Slack node, Slack sends 50 messages. Each time, expressions like `{{ $json.name }}` resolve to the current item's value.

A few nodes are exceptions and work on the whole list at once, for example the Code node in **Run Once for All Items** mode, Aggregate, Summarize and Merge. Some database and API operations also run only once regardless of item count. The node's documentation says so when that's the case.

NOTE

Sub-nodes attached to AI nodes (chat models, memory, tools) behave differently: an expression in a sub-node always resolves to the **first** item.

8.4 Inspecting data

Every node's input and output panel has three views:

- **Table:** rows and columns, like a spreadsheet. Nested data shows in bold; click to expand.
- **JSON:** the exact structure. Best when you need precise paths.
- **Schema:** a list of field names and types, without values. Best for dragging fields into parameters.

Above the panel, n8n shows how many items there are, for example *3 items*. Check this number whenever results surprise you.

8.5 Files: binary data

Binary data is anything that isn't text: a PDF invoice, a photo, a spreadsheet file, an audio clip. n8n keeps files in the item's `binary` section, under a name such as `data` or `attachment_0`:

JSON

```
{
  "json": { "subject": "Your invoice" },
  "binary": {
    "data": {
      "fileName": "invoice-1042.pdf",
      "mimeType": "application/pdf",
      "fileExtension": "pdf"
    }
  }
}
```

The **MIME type** describes the kind of file (`image/png` , `application/pdf`). In the output panel, binary items show a **View** button so you can preview or download them.

NODE	WHAT IT DOES WITH FILES
Extract From File	Reads CSV, XLSX, PDF, HTML, JSON, ICS, RTF or text into JSON fields
Convert to File	Turns JSON into a CSV, XLSX, HTML, ICS, JSON or text file
Read/Write Files from Disk	Reads or saves files on a self-hosted server
Compression	Zips and unzips files
Edit Image	Resizes, crops and annotates images
HTTP Request	Downloads files when the response is a file

When a node like Gmail or Google Drive needs a file, it asks for the **binary property name** (often `data`). That's simply which file slot to use.

TIP

Email nodes can download attachments as binary data, usually in numbered slots such as `attachment_0` . Look at the node's output panel to see the exact names, then use that name wherever the next node asks for a binary property.

8.6 Splitting and combining items

Sometimes data arrives in the wrong shape. Two nodes fix most cases:

- **Split Out** turns one item containing a list into many items. One order with five products becomes five items, one per product.
- **Aggregate** does the reverse: many items become one item with a list, handy for building a single summary email.

Others you'll use: **Summarize** (totals and counts, like a pivot table), **Sort, Limit** (keep only the first n), and **Remove Duplicates**.

TURN ONE LIST INTO MANY ITEMS

- 1 **Start with one item** that has an `orders` array (pin some sample data as described below).
- 2 **Add Split Out.**
Set **Fields To Split Out** to `orders`.
- 3 **Execute the step.**
The output now shows one item per order, ready for per-order processing.

8.7 Item linking

n8n remembers which input item produced each output item. This **item linking** (also called *paired items*) is what lets `{{ $('Get Customer').item.json.name }}` find the right customer for the current order, even several nodes later. You rarely need to think about it. If you see an error about paired items, it usually means a Code node changed the number of items; Chapter 10 explains how to fix that.

8.8 Pinning and mocking data

Pinning freezes a node's output so later runs reuse it instead of calling the real service. Open a node after it has run and click the **pin** icon in the output panel, or select the node and press **P**.

You can also **mock** data: click the edit (pencil) icon on an output panel and type or paste your own JSON. This is perfect for building a workflow before the real data source exists.

☆ IMPORTANT

Pinned data is for building only. Production executions ignore pins and always fetch fresh data, so a published workflow behaves correctly even if you forget to unpin.

- ☐ I can read a JSON object and follow a path like `address.city`
- ☐ I know n8n data is a list of items, each with `json` and maybe `binary`
- ☐ I check the item count when results surprise me
- ☐ I know Split Out and Aggregate change the number of items
- ☐ I can pin and mock data while building

Expressions: Little Formulas, Big Power

An **expression** is a small formula inside a node's parameter that fills in a value when the workflow runs. "Send this email to *whoever filled in the form*." "Name the file *today's date*." If you've used formulas in a spreadsheet, you already understand the idea. This chapter teaches you to read and write them with confidence.

IN THIS CHAPTER

- How to switch a field to expression mode and map by dragging
- Reaching data from the current item and from earlier nodes
- Built-in variables: dates, workflow and execution details
- Handy functions for text, numbers and lists
- Working with dates using Luxon
- Getting help from AI and fixing common errors

9.1 Mapping by dragging

The easiest way to use data from an earlier node: open a node, find the field in the **Input** panel, and **drag it** into a parameter. n8n writes the expression for you.

9.2 Expression mode

Every parameter can be **Fixed** or **Expression**. Click **Expression** above a field (or type `=` in an empty one) and the field turns into an editor that shows a live **preview** of the result underneath. Anything inside `{{ }}` is calculated; anything outside is plain text:

TEXT

Hello {{ \$json.firstName }}, your order #{{ \$json.orderId }} has shipped!

 TIP

Watch the preview as you type. If it shows the value you expect for the first item, you're nearly always right. If it says `undefined`, the path is wrong: check spelling and capitals.

9.3 Expressions cheat sheet

Expressions live inside double curly braces and can contain small bits of JavaScript.

EXPRESSION	GIVES YOU
<code>{{ \$json.email }}</code>	The <code>email</code> field of the current item
<code>{{ \$json.name.toUpperCase() }}</code>	The name in capital letters
<code>{{ \$('Google Sheets').item.json.total }}</code>	A field from a specific earlier node, by name
<code>{{ \$now.toFormat('yyyy-MM-dd') }}</code>	Today's date, like <code>2026-10-15</code>
<code>{{ \$today.minus(7, 'days') }}</code>	The date one week ago
<code>{{ \$json.price * 1.08 }}</code>	Arithmetic: price plus 8% tax
<code>{{ \$json.status === 'paid' ? 'Thanks!' : 'Reminder' }}</code>	Choose text based on a condition

 TIP

Not sure how to write an expression? Click **Ask AI** or the AI assistant in the expression editor and describe what you want: “the first name from the `full_name` field.”

9.4 Reaching your data

WRITE THIS	TO GET
<code>\$json.field</code>	A field of the current item
<code>\$json.address.city</code>	A nested field
<code>\$json.tags[0]</code>	The first entry of a list
<code>\$json['First Name']</code>	A field whose name has spaces
<code>\$('Node Name').item.json.field</code>	The matching item from an earlier node
<code>\$('Node Name').first().json.field</code>	The first item from an earlier node
<code>\$('Node Name').all()</code>	Every item from an earlier node, as a list

<code>\$input.first().json.field</code>	The first item arriving at this node
<code>\$input.all().length</code>	How many items arrived

`.item` is the clever one: it follows item linking (Chapter 8) to find the item that belongs to the one you're processing now. Use `.first()` when an earlier node produced a single item, such as a settings lookup.

9.5 Built-in variables

VARIABLE	GIVES YOU
<code>\$now</code>	The current date and time
<code>\$today</code>	Today at midnight
<code>\$workflow.name</code> / <code>\$workflow.id</code>	This workflow's name and ID
<code>\$workflow.active</code>	Whether it's published
<code>\$execution.id</code>	This run's ID, handy in error messages
<code>\$execution.mode</code>	<code>test</code> (you clicked a button) or <code>production</code>
<code>\$itemIndex</code>	The position of the current item, from 0
<code>\$vars.NAME</code>	A custom variable (paid plans)

9.6 Helper functions

n8n adds its own conveniences on top of JavaScript:

FUNCTION	EXAMPLE	RESULT
<code>\$if()</code>	<code>{{ \$if(\$json.vip, 'Gold', 'Standard') }}</code>	Picks one of two values
<code>\$ifEmpty()</code>	<code>{{ \$ifEmpty(\$json.nickname, \$json.name) }}</code>	A fallback when a value is empty
<code>\$jmespath()</code>	<code>{{ \$jmespath(\$json, 'orders[*].id') }}</code>	Queries JSON with JMESPath

9.7 Functions for text, numbers and lists

Add these after a value with a dot. Type a dot in the expression editor to see suggestions.

FUNCTION	ON	EXAMPLE	GIVES
<code>.toTitleCase()</code>	Text	<code>"ana lópez".toTitleCase()</code>	Ana López
<code>.extractEmail()</code>	Text	<code>"Mail: a@b.com".extractEmail()</code>	a@b.com
<code>.extractDomain()</code>	Text	<code>"a@shop.com".extractDomain()</code>	shop.com

<code>.removeTags()</code>	Text	<code>"Hi".removeTags()</code>	Hi
<code>.removeMarkdown()</code>	Text	<code>"**Hi**".removeMarkdown()</code>	Hi
<code>.isEmpty()</code>	Text, list	<code>"".isEmpty()</code>	true
<code>.toNumber()</code>	Text	<code>"42".toNumber()</code>	42
<code>.urlencode()</code>	Text	<code>"a b".urlencode()</code>	a%20b
<code>.toDateTime()</code>	Text	<code>"2026-10-16".toDateTime()</code>	A date you can format
<code>.sum()</code>	List	<code>[1,2,3].sum()</code>	6
<code>.average()</code>	List	<code>[2,4].average()</code>	3
<code>.unique()</code>	List	<code>[1,1,2].unique()</code>	[1,2]
<code>.pluck()</code>	List	<code>\$json.people.pluck('email')</code>	Just the emails
<code>.first()</code> / <code>.last()</code>	List	<code>[5,6,7].last()</code>	7
<code>.join()</code>	List	<code>['a','b'].join(',')</code>	a, b
<code>.chunk()</code>	List	<code>[1,2,3,4].chunk(2)</code>	[[1,2],[3,4]]

9.8 Dates and times with Luxon

n8n's `$now` and `$today` are **Luxon** date objects. Luxon is a popular JavaScript date library, and it makes date maths readable:

EXPRESSION	RESULT (EXAMPLE)
<code>{{ \$now.toFormat('d LLLL yyyy') }}</code>	16 October 2026
<code>{{ \$now.toFormat('cccc') }}</code>	Friday
<code>{{ \$now.toFormat('HH:mm') }}</code>	07:00
<code>{{ \$now.plus(1, 'day').toFormat('yyyy-MM-dd') }}</code>	Tomorrow's date
<code>{{ \$today.startOf('month').toISO() }}</code>	The first of this month
<code>{{ \$json.created.toDateTime().toFormat('d LLL') }}</code>	Format a date from your data

Common format codes: `yyyy` year, `MM` month number, `LLL` short month name, `LLLL` full month name, `dd` day, `cccc` weekday, `HH` hour (24h), `mm` minutes.

NOTE

Dates use the workflow's time zone if one is set in workflow settings, otherwise the instance time zone. If a time is an hour out, that's usually the reason.

9.9 Expressions or nodes?

You can do a lot inside expressions, but long formulas are hard to read. A good rule:

- **Short and specific** (format a name, pick a field): use an expression.
- **Reshaping many fields**: use **Edit Fields (Set)**.
- **Logic across many items** (totals, grouping, complex rules): use **Aggregate**, **Summarize**, or a **Code** node (Chapter 10).

9.10 Worked example: a friendly order summary

Given an item like `{ "firstName": "ana", "items": [ {"price": 12}, {"price": 30} ], "date": "2026-10-16" }`, this expression:

TEXT

```
Hi {{ $json.firstName.toUpperCase() }}, your {{ $json.items.length }} items
total ${{ $json.items.pluck('price').sum() }}, ordered on
{{ $json.date.toDateTime().toFormat('d LLLL') }}.
```

produces *Hi Ana, your 2 items total \$42, ordered on 16 October.*

PROMPT

I'm writing an n8n expression. My item looks like this: [paste the JSON from the input panel]. I want [describe the result, e.g. "the customer's first name in capitals, or 'friend' if it's empty"]. Give me the expression inside {{ }} and explain each part.

9.11 Common expression errors

MESSAGE OR SYMPTOM	CAUSE	FIX
Preview shows <code>undefined</code>	Wrong path or capitalisation	Drag the field in instead of typing
Preview reports invalid syntax	A missing bracket or quote	Count your <code>(</code> , <code>)</code> , <code>'</code> and <code>}}</code>
"Can't get data for expression"	The referenced node hasn't run yet	Execute the earlier node first
Paired item error	Item links broken by a Code node	Use <code>.first()</code> or fix the Code node (Chapter 10)
Shows <code>[object Object]</code>	You inserted a whole object into text	Pick a field, or wrap it in <code>JSON.stringify()</code>

- ☐ I can switch a field to expression mode and read the preview
- ☐ I can reach a field from an earlier node with `$( 'Node Name' )`
- ☐ I can format today's date with Luxon
- ☐ I know a few text and list functions
- ☐ I know when to use a node instead of a long expression

The Code Node: JavaScript and Python

Most workflows never need code. But sometimes you need one small calculation that no node offers, or a tricky reshaping of data. The **Code** node lets you write a few lines of JavaScript or Python as a step. You don't need to be a programmer: with the examples here, and an AI assistant to help, a beginner can use it safely.

IN THIS CHAPTER

- When to reach for code, and when not to
- The two modes: once for all items, or once for each item
- What your code receives and must return
- Five copy-paste JavaScript recipes
- Python in n8n and how it differs
- Limits on Cloud and self-hosted, and getting AI help

10.1 When to use code

USE A CODE NODE WHEN...	USE OTHER NODES WHEN...
A calculation has many steps	You only need to rename or add fields (Edit Fields)
You need custom logic across all items	You need to branch (If, Switch)
No node does the transformation	A node or expression already does it
You're parsing an odd text format	You need HTTP calls or files (HTTP Request, Read/Write Files)

NOTE

The Code node can't make HTTP requests or read files on disk. Use the HTTP Request and Read/Write Files from Disk nodes for those, and pass their results into your code.

10.2 The two modes

MODE	RUNS	YOU USE
Run Once for All Items (default)	One time, with every item	<code>\$input.all()</code>
Run Once for Each Item	Once per item	<code>\$json</code> or <code>\$input.item</code>

Choose **Each Item** for simple per-row changes. Choose **All Items** when you need to look across rows: totals, sorting, grouping, or comparing.

10.3 What goes in and what comes out

Your code receives items and must **return** items. Each returned item is an object with a `json` key (n8n will add it if you forget, and wrap a single object in a list for you). This minimal example, in **Run Once for All Items** mode, adds a field to every item:

ADD A FIELD TO EVERY ITEM

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

for (const item of items) {
  item.json.fullName = `${item.json.first} ${item.json.last}`;
}

return items;
```

The same thing in **Run Once for Each Item** mode is even shorter:

PER-ITEM VERSION

```
const j = $input.item.json;
return { json: { ...j, fullName: `${j.first} ${j.last}` } };
```

The `...j` copies all existing fields, then `fullName` is added.

10.4 Five JavaScript recipes

1. Total an order list (All Items mode). Returns one item with the sum:

JAVASCRIPT

```
const total = $input.all()
  .reduce((sum, item) => sum + Number(item.json.amount || 0), 0);

return [{ json: { total, count: $input.all().length } }];
```

2. Keep only recent items (All Items mode). Drops anything older than seven days:

JAVASCRIPT

```
const cutoff = Date.now() - 7 * 24 * 60 * 60 * 1000;

return $input.all().filter(
  (item) => new Date(item.json.date).getTime() >= cutoff
);
```

3. Group by a field (All Items mode). Turns orders into one item per customer:

JAVASCRIPT

```
const groups = {};
for (const { json } of $input.all()) {
  groups[json.customer] ??= { customer: json.customer, orders: 0 };
  groups[json.customer].orders += 1;
}
return Object.values(groups).map((g) => ({ json: g }));
```

4. Clean up messy text (Each Item mode):

JAVASCRIPT

```
const raw = $input.item.json.phone || '';
const digits = raw.replace(/\D/g, '');
return { json: { ...$input.item.json, phone: digits } };
```

5. Build an HTML table for an email (All Items mode):

JAVASCRIPT

```
const rows = $input.all().map(({ json }) =>
  `<tr><td>${json.name}</td><td>${json.status}</td></tr>`
).join('');

return [{ json: { html: `<table>${rows}</table>` } }];
```

 TIP

Use `console.log(something)` while building. The output appears in your **browser's developer console** (press **F12** and choose **Console**), which is handy for peeking inside variables.

10.5 Built-ins you can use in code

Everything from Chapter 9 works in JavaScript code too: `$input`, `$json`, `$('Node Name').all()`, `$now`, `$today`, `$workflow`, `$execution` and `$vars`. Type `$` in the editor to see suggestions.

10.6 Python

Choose **Python** from the language option on the Code node. n8n runs real Python (“native Python”) through its **task runners**, separate processes that execute code safely away from the main n8n program. It works a little differently from JavaScript:

- In **Run Once for All Items** mode, your items are in `_items`. In **Run Once for Each Item** mode, the current item is `_item`.
- Use square brackets to reach fields: `item["json"]["name"]`, not `item.json.name`.
- Other n8n built-ins (like `$('Node')`) aren't available in native Python.

PYTHON, RUN ONCE FOR ALL ITEMS

```
for item in _items:
    name = item["json"].get("name", "")
    item["json"]["name_upper"] = name.upper()

return _items
```

 NOTE

On n8n Cloud, Python Code nodes can't import any libraries, even from Python's standard library. On self-hosted n8n you can allow specific standard-library and third-party modules through the task runner configuration. The older in-browser Python (Pyodide) was removed in n8n 2.0, so old Python examples using `item.json.field` need updating.

10.7 Libraries in JavaScript

On n8n Cloud, the JavaScript Code node can use two extra modules: Node.js's `crypto` (for hashing and random IDs) and the `moment` date library. Self-hosted users can allow other built-in and npm modules through environment settings (see Chapter 26).

10.8 Keeping items linked

When your code returns the **same number of items in the same order**, n8n keeps item linking intact, and later nodes can still use `$('#Earlier Node').item`. When you return fewer or more items (filtering, grouping, splitting), the links may break. If a later node complains about paired items, either reference earlier data with `.first()` or `.all()`, or include the fields you need in your returned items so later nodes don't have to look back.

10.9 Let AI write it

The Code node has AI help built in on n8n Cloud: describe what you want, and it drafts the code. Any chat assistant works too:

PROMPT

Write an n8n Code node in JavaScript, mode "Run Once for All Items".
Input items look like this: [paste one item's JSON]. I want the output to be [describe it]. Return an array of objects with a "json" key. Keep it short and add a comment on each line explaining what it does.

⚠ WARNING

Read AI-written code before you run it on real data. Test with pinned sample data first, and check the output panel item by item.

10.10 Common Code node errors

ERROR	CAUSE	FIX
"Code doesn't return items properly"	Returned something that isn't a list of objects	Return <code>[{ json: {...} }]</code> or the items list
"Cannot read properties of undefined"	A field you used doesn't exist on some item	Use <code>item.json.x ?? ''</code> for a default
<code>\$input.item</code> is undefined	Wrong mode	Use <code>\$input.all()</code> in All Items mode
Cannot find module	Library not allowed on this instance	Use built-ins, or allow the module (self-hosted)
Python <code>KeyError</code>	A field is missing	Use <code>item["json"].get("field")</code>

- ☐ I know the difference between the two modes
- ☐ I can return items in the right shape
- ☐ I've tried at least one recipe with pinned data
- ☐ I know Python uses `_items` and square brackets
- ☐ I check AI-written code before trusting it

Core Nodes: The Toolkit You'll Use Every Day

App nodes talk to other services; **core nodes** are the logic and plumbing that join everything together. Learn these ten or so and you can build almost any workflow. This chapter explains each one with a small example.

IN THIS CHAPTER

- Edit Fields (Set), Filter, If and Switch
- Merge and Loop Over Items
- HTTP Request, including pagination and cURL import
- Wait, and the smaller helpers
- A reference table of every everyday core node

11.1 Edit Fields (Set)

Edit Fields (often still called the Set node) adds, changes or removes fields. It has two modes: **Manual Mapping**, where you add fields one by one in the interface, and **JSON Output**, where you write the JSON yourself.

- Turn on **Keep Only Set Fields** (or set **Include in Output** accordingly) to drop everything you didn't list, which keeps later nodes tidy.
- Each field has a type (String, Number, Boolean, Array, Object). Choose the right one so later comparisons work.

TIP

Put an Edit Fields node right after a trigger to rename messy field names (`Your E-mail Address`) into clean ones (`email`). Every later node becomes easier to read.

11.2 Filter

Filter keeps only items that match your conditions and silently drops the rest. Use it to skip unneeded work early, which also saves executions downstream. Example: keep orders where

```
{{ $json.total }} is greater than 100 .
```

11.3 If

If sends each item down a **true** or **false** output. Conditions compare values by type: String (contains, starts with, is empty...), Number (greater than...), Date & Time (is after...), Boolean, Array (contains, length...) and Object. Combine several conditions with **AND** (all must match) or **OR** (any may match).

11.4 Switch

Switch is an If with many outputs, like a railway junction. In **Rules** mode, you add one rule per output, for example “category is *billing*,” “category is *technical*.” Useful options:

- **Fallback Output:** where items go if no rule matches. Add an extra output so nothing disappears silently.
- **Send data to all matching outputs:** otherwise an item only goes to the first matching rule.
- **Ignore Case:** treat `Billing` and `billing` the same.

Expression mode lets an expression return the output number directly.

11.5 Merge

Merge waits for two (or more) inputs and combines them. Choose a mode:

MODE	DOES	LIKE
Append	Lists all items from input 1, then input 2	Stacking two piles
Combine › Matching Fields	Joins items where a field matches (e.g. <code>email</code>)	A spreadsheet lookup
Combine › Position	Pairs item 1 with item 1, item 2 with item 2	Zippping two lists
Combine › All Possible Combinations	Every item with every other item	A grid
SQL Query	Write SQL over the inputs	A database query
Choose Branch	Waits for both, outputs one	A gate

When matching fields, choose what to keep: **Keep Matches** (inner join), **Keep Everything** (outer join), **Enrich Input 1** (keep all of input 1, add matches from input 2) and so on.

NOTE

A common pattern is If → two branches → Merge. Merge waits for both branches, so the workflow continues once, with all the data, after the paths split.

11.6 Loop Over Items (Split in Batches)

Usually you **don't need a loop**: nodes already process every item. Use **Loop Over Items** when you must handle items in smaller groups, typically to respect an API's **rate limit** (how many requests per minute it accepts).

PROCESS 500 CONTACTS, 10 AT A TIME

- 1 **Add Loop Over Items** after the node producing the contacts. Set **Batch Size** to `10`.
- 2 **Connect the loop output** to the work you want to do (for example an HTTP Request), then a **Wait** node of 2 seconds.
- 3 **Connect the Wait node back** to the Loop Over Items input. That closes the loop.
- 4 **Connect the done output** to whatever should happen at the end. It receives all processed items once every batch has finished.

You can also build a “loop until” by connecting a later node back to an earlier one with an If node deciding when to stop.

11.7 HTTP Request

HTTP Request calls any web API, so it's your escape hatch when n8n has no dedicated node. The key settings:

SETTING	WHAT TO PUT
Method	<code>GET</code> to read, <code>POST</code> to create, <code>PUT</code> / <code>PATCH</code> to update, <code>DELETE</code> to remove
URL	The API endpoint from the service's docs
Authentication	A predefined credential for a known app, or a generic one (header, bearer, basic, OAuth2)
Send Query Parameters	Values after <code>?</code> in the URL
Send Headers	Extra headers the API asks for
Send Body	JSON, form data, or a binary file for POST/PUT

Useful options: **Batching** (items per batch plus a pause, for rate limits), **Pagination** (fetch all pages automatically, either by updating a parameter each request or by following a “next URL” in the response), **Response › Never Error** (don't fail on 4xx/5xx codes), and **Timeout**.

TIP

API docs often show a `curl` command. Click **Import cURL** in the HTTP Request node, paste it, and n8n fills in method, URL, headers and body for you. Move any secret key into a credential afterwards.

TIP

Choosing **Predefined Credential Type** lets you reuse, say, your Notion credential to call a Notion API endpoint the Notion node doesn't cover.

11.8 Wait

Wait pauses a workflow. Its modes are **After Time Interval** (wait 10 minutes), **At Specified Time** (wait until Monday 9am), **On Webhook Call** (wait until something calls a unique “resume” URL, available as `$execution.resumeUrl`) and **On Form Submitted** (wait until someone completes a form). Long waits don't hold up your server: n8n saves the execution and resumes it later.

11.9 Everyday helpers

NODE	USE IT TO
Aggregate	Combine many items into one item with lists
Split Out	Turn a list inside one item into many items
Summarize	Count, sum or average, grouped by a field
Sort / Limit	Order items; keep only the first <i>n</i>
Remove Duplicates	Drop repeats in this run, or items seen in previous runs
Compare Datasets	Find what's new, changed or missing between two lists
Date & Time	Format, add to, or compare dates
Crypto	Hash values, generate random strings
HTML	Extract content from web pages with CSS selectors
Respond to Webhook	Send a custom reply to a Webhook caller
Execute Sub-workflow	Call another workflow (Chapter 14)
Stop and Error	Fail on purpose with a clear message (Chapter 17)
No Operation, do nothing	A placeholder that keeps the canvas tidy
Data Table	Read and write n8n's built-in tables (Chapter 15)

 TIP

Remove Duplicates has a mode called **Remove Items Processed in Previous Executions**. It remembers what it has seen across runs, which is perfect for “only alert me about new RSS posts” without a database.

- ☐ I can reshape data with Edit Fields
- ☐ I can branch with If and Switch, and rejoin with Merge
- ☐ I know when I need Loop Over Items (and when I don't)
- ☐ I can call an API with HTTP Request and import a cURL command
- ☐ I know Remove Duplicates can remember previous runs

Triggers: How Workflows Start

Every workflow begins with a trigger. Picking the right one is the first design decision you make, and it affects how quickly your workflow reacts and how many executions it uses. This chapter covers each kind, with set-up steps for the most important.

IN THIS CHAPTER

- The trigger types at a glance
- Schedules, including Cron expressions
- Webhooks: URLs, security and responses
- Forms, multi-page forms and chat
- App triggers: instant versus polling
- Test URLs versus production URLs

TRIGGER	STARTS WHEN...	EXAMPLE
Schedule Trigger	A time arrives: every hour, every Monday at 9...	Weekly report
Webhook	Another app sends data to n8n's unique URL	Payment received in Stripe
n8n Form Trigger	Someone submits a form n8n hosts for you	Event sign-up
App triggers	Something happens in an app (new email, new row, new file)	New Gmail message with a label
Chat Trigger	Someone sends a message in a chat window	A chatbot for your website
Execute Workflow Trigger	Another workflow calls this one	Reusable "send notification" sub-workflow
Error Trigger	Another workflow fails	Alert yourself on failure
Manual	You click Execute workflow	Testing
MCP Server Trigger	An AI app calls one of its tools	Chapter 22

12.1 Schedule Trigger

Choose a **Trigger Interval**: Seconds, Minutes, Hours, Days, Weeks, Months, or **Custom (Cron)**. Each interval asks for the details you'd expect, such as **Trigger at Hour** and **Trigger at Minute**, or the weekdays for a weekly schedule. You can add several rules to one trigger.

Cron is a compact way to write schedules. n8n's Cron field accepts an optional seconds column at the front, which is why you may see six positions:

CRON EXPRESSION	MEANS
0 9 * * 1-5	9:00 every weekday
*/15 * * * *	Every 15 minutes
0 8 1 * *	8:00 on the 1st of each month
0 0 18 * * 5	18:00 every Friday (with seconds first)

The five standard positions are minute, hour, day of month, month and day of week. Sites like crontab.guru explain any expression in plain English.

☆ IMPORTANT

Schedules only run when the workflow is **published**, and they use the workflow's time zone (set in workflow settings) or, failing that, the instance time zone. Self-hosted instances default to New York time.

 TIP

Every schedule run is a production execution. A trigger every minute is about 43,000 runs a month, far beyond a Starter plan. Every 15 minutes is about 2,900. Pick the slowest schedule that does the job.

12.2 Webhook

A **webhook** is a web address that another service calls when something happens. The Webhook node gives you that address.

SET UP A WEBHOOK

- 1 Add a Webhook node.**
Choose the **HTTP Method** the other service will use, usually `POST`.
- 2 Optionally set a Path**, such as `new-order`, instead of the random one. Paths can include parameters like `/order/:id`.
- 3 Choose Authentication.**
Basic auth, header auth or JWT auth stop strangers from triggering your workflow. Use one for anything public.
- 4 Choose Respond.**
Immediately replies “Workflow got started”; **When Last Node Finishes** replies with the final data; **Using ‘Respond to Webhook’ Node** lets you build a custom reply.
- 5 Copy the Test URL**, click **Listen for test event**, and send a test from the other service. The data appears in the output.
- 6 Publish**, then give the other service the **Production URL**.

Useful options include **IP(s) Allowlist** (only accept calls from listed addresses), **Ignore Bots**, **Allowed Origins (CORS)** for calls from web pages, and **Binary Property** to receive files. The maximum payload is 16 MB by default.

12.3 Test URLs versus production URLs

Webhook and Form triggers each have **two** addresses:

- **Test URL**: works while you click **Listen for test event** in the editor, so you can see data arrive.
- **Production URL**: works once the workflow is **published**.

On self-hosted n8n they differ only in the path: test URLs contain `/webhook-test/` and production URLs contain `/webhook/`.

The most common beginner puzzle, “my webhook works in testing but not for real,” is almost always because the workflow isn’t published, or the other app is still using the test URL.

12.4 n8n Form Trigger

n8n can host a simple web form for you, with no website needed.

BUILD A SIGN-UP FORM

- 1 Add an n8n Form Trigger.**
Give it a **Form Title** and **Form Description**.
- 2 Add form elements.**
Types include Text, Email, Number, Date, Dropdown, Checkboxes, Radio Buttons, Textarea, File, Password, Hidden Field and Custom HTML. Tick **Required Field** where needed.
- 3 Choose Respond When.**
Form Is Submitted confirms straight away; **Workflow Finishes** waits until the workflow has run.
- 4 Test it.**
Click **Execute step**; n8n opens the form. Submit it and the answers appear as fields named after each element.
- 5 Publish and share the production URL.**

For **multi-page forms**, add **n8n Form** nodes after the trigger. Each is another page, and the last can show a completion screen, redirect to a web page, show your own text, or return a file. You can put If nodes between pages to show different questions depending on earlier answers.

TIP

Options let you hide the “Form automated with n8n” line, rename the submit button, set a custom path, and restyle the form with CSS.

12.5 Chat Trigger

The **Chat Trigger** starts a workflow from a chat message, usually feeding an AI Agent (Chapter 20). Turn on **Make Chat Publicly Available** to get a shareable chat page. **Mode** is either **Hosted Chat** (n8n’s ready-made chat page, set up with options for title, placeholder and welcome message) or **Embedded Chat** (your own website calls the workflow, for example with n8n’s chat widget). Authentication can be none, basic auth, or **n8n User Auth** (only people logged in to your n8n).

12.6 App triggers: instant or polling?

App triggers come in two flavours:

- **Instant (webhook-based)**: the app tells n8n immediately. Examples: Telegram, Stripe, GitHub. Needs a public HTTPS address, so it won't work on a laptop without a tunnel.
- **Polling**: n8n checks the app on a timer, set by **Poll Times**. Examples: Gmail Trigger, Google Sheets Trigger. Each check that finds something new is an execution.

WARNING

Telegram allows only one webhook per bot. If you test a Telegram Trigger while the same workflow is published, the test URL replaces the production one, and the published workflow stops receiving messages until you publish again.

12.7 Other triggers

- **Execute Sub-workflow Trigger** (shown as “When Executed by Another Workflow”): Chapter 14.
- **Error Trigger**: Chapter 17.
- **RSS Feed Trigger**: new items in a feed.
- **Local File Trigger** (self-hosted): a file changes in a folder.

- ☐ I can choose between schedule, webhook, form, chat and app triggers
- ☐ I know schedules use the workflow or instance time zone
- ☐ I've secured any public webhook with authentication
- ☐ I give other services the production URL, not the test URL
- ☐ I know polling triggers use executions every time they find something

Connecting Your Apps: Credentials and OAuth

To read your email or post to your Slack, n8n needs permission. That permission is stored as a **credential**. This chapter explains the main kinds, walks through the trickiest one (Google on a self-hosted server), and shows how to call any API.

IN THIS CHAPTER

- What credentials are and where they live
- OAuth versus API keys
- Setting up Google OAuth on self-hosted n8n
- Restricting which domains a credential may be used with
- Calling any API with HTTP Request
- Good security habits

13.1 Credentials

A **credential** stores the sign-in for an app so nodes can use it. Create them from a node (**Credential** > **Create new credential**) or from **Credentials** in the sidebar. There are two main kinds:

- **OAuth** (“Sign in with Google,” “Connect to Slack”): click a button and approve in a pop-up. On n8n Cloud, most of these work instantly.
- **API key or token**: copy a secret key from the app’s developer settings and paste it in.

When you save a credential, n8n tests it to confirm it works. Credentials are encrypted in n8n’s database with your instance’s encryption key, and their secret values are never shown again after saving.

13.2 How OAuth works, in plain English

OAuth is the “valet key” of the internet. Instead of giving n8n your Google password, you sign in to Google, and Google hands n8n a limited key that can do only what you approved (for example “read and send Gmail”). You can revoke that key at any time from your Google account without changing your password.

The flow has one fiddly part: the app must know where to send you back after you approve. That address is the **OAuth Redirect URL**, shown in every n8n OAuth credential. It must match exactly what’s registered with the app.

13.3 Managed versus custom OAuth

On **n8n Cloud**, many popular apps offer **Managed OAuth2**: just click **Sign in with Google** and you’re done. For Google, that covers Gmail, Sheets, Drive, Calendar, Docs, Slides, Contacts, Tasks, Chat and YouTube, plus their triggers.

On **self-hosted** n8n (and for apps without managed OAuth), you create your own **OAuth app** in the provider’s developer console. Each credential page in n8n links to step-by-step instructions.

13.4 Walkthrough: Google OAuth on self-hosted n8n

CREATE A GOOGLE OAUTH CREDENTIAL

- 1 Create a Google Cloud project.**
Go to console.cloud.google.com, open the project dropdown and choose **New project**. Name it “n8n” and select **Create**.
- 2 Enable the APIs you need.**
Go to **APIs & Services** > **Library**, search for, say, **Gmail API**, and select **Enable**. Repeat for Sheets, Drive and so on.
- 3 Configure the consent screen.**
Go to **APIs & Services** > **OAuth consent screen** and select **Get started**. Enter an app name and support email. For **Audience**, choose **Internal** (Workspace users only) or **External** (any Google account). Accept the policy and create it.
- 4 Add your domain.**
Under **Branding** > **Authorized domains**, add your n8n domain (for example `example.com`).

- 5 Create the client.**
Go to **APIs & Services > Credentials**, choose **+ Create credentials > OAuth client ID**, and set **Application type** to **Web application**.
- 6 Paste the redirect URL.**
In n8n, create a Google credential (for example “Gmail OAuth2 API”) and copy its **OAuth Redirect URL**. Paste it under **Authorized redirect URIs** in Google, then select **Create**.
- 7 Copy the ID and secret back.**
Paste Google’s **Client ID** and **Client Secret** into the n8n credential, select **Sign in with Google**, approve, and **Save**.

⚠ WARNING

With an **External** audience, your Google app starts in **Testing** mode: only Google accounts you add as **test users** can sign in. Everyone else sees “access denied.” Add yourself as a test user on the consent screen settings.

13.5 Restricting where a credential can be used

Many credentials have an **Allowed HTTP Request Domains** field. Set it to **Specific Domains** (for example `api.github.com`) so the credential can’t be sent to any other site from an HTTP Request node, or **None** to block HTTP Request use entirely. This protects you if someone edits a workflow carelessly.

13.6 Fixed and end-user credentials

For OAuth credentials, some plans let you choose a **credential type**: a **Fixed credential** (the same account whatever happens, the default) or an **End-user credential** (each person who triggers the workflow uses their own account, a preview feature on higher plans).

13.7 When there’s no node: HTTP Request

Almost every online service has an **API**. If n8n has no dedicated node for it, use the **HTTP Request** node with the service’s documentation. Many API docs show a `curl` command; n8n’s HTTP Request node has **Import cURL**, which fills in everything for you.

CALL AN API WITH A KEY

- 1 Find the API docs.**
Note the endpoint URL and how it wants the key, usually a header such as `Authorization: Bearer YOUR_KEY` or `X-API-Key: YOUR_KEY`.

2

Create a generic credential.

In the HTTP Request node choose **Authentication > Generic Credential Type**, then **Header Auth** (or **Bearer Auth**). Enter the header name and value.

3

Set method and URL, add any query parameters, and click **Execute step.**

4

Inspect the response.

If it fails, read the status code: 401 means the key is wrong, 403 not allowed, 404 wrong URL, 429 too many requests.

13.8 Sharing credentials

On plans with projects, a credential belongs to a project or to your personal space. Everyone in a project can **use** its credentials in workflows without seeing the secret values. Credential sharing with specific people requires a paid plan.

13.9 Security habits

- Name credentials clearly, for example “Gmail: personal” and “Gmail: business.”
- Give each credential the minimum access it needs.
- Never paste keys into Code nodes or text fields; always use credentials.
- Remove credentials you no longer use.
- Revoke an app’s access at the provider (Google, Slack...) if a credential might have leaked, then create a fresh one.

- ☐ I’ve created at least one OAuth and one API-key credential
- ☐ I know where to find the OAuth Redirect URL
- ☐ On self-hosted, I’ve added myself as a Google test user
- ☐ I’ve restricted credential domains where it makes sense
- ☐ No secrets live in my Code nodes or text fields

Sub-workflows: Build Once, Reuse Everywhere

As you build more workflows, you'll notice the same steps appearing again and again: "send a Slack alert," "look up a customer," "log this to a sheet." A **sub-workflow** is a workflow that other workflows call, like a function in programming. Build it once, call it from anywhere, fix it in one place.

IN THIS CHAPTER

- Why sub-workflows make life easier
- Creating a sub-workflow with defined inputs
- Calling it with Execute Sub-workflow
- How data passes in and out
- Turning part of an existing workflow into a sub-workflow

14.1 Why bother?

- **Less repetition.** Change your alert format once, and every workflow that uses it updates.
- **Tidier canvases.** A 60-node workflow becomes a readable 10-node workflow calling helpers.
- **Lower memory use.** Big jobs split into smaller runs.
- **Cheaper.** Sub-workflow executions don't count towards your plan's monthly execution limit.
- **AI tools.** An AI agent can call a sub-workflow as a tool (Chapter 20).

14.2 Create the sub-workflow

BUILD A REUSABLE "NOTIFY ME" WORKFLOW

- 1 **Create a new workflow** called "Notify me".

2 Add the Execute Sub-workflow Trigger.

In the trigger list it's titled **When Executed by Another Workflow**.

3 Define the inputs.

Set **Input data mode** to **Define using fields below**, and add `title` (String) and `message` (String). Other modes are **Define using JSON example** and **Accept all data**.

4 Add the work.

For example a Slack node posting `*{{ $json.title }}*` and `{{ $json.message }}` to your alerts channel.

5 Make sure it has no errors.

A parent workflow can't call a sub-workflow that contains errors.

TIP

In the sub-workflow's settings, **This workflow can be called by** controls who may use it: for example only workflows in the same project, or specific workflows.

14.3 Call it from another workflow

CALL "NOTIFY ME"

1 Add an Execute Sub-workflow node where you want the notification.

2 Choose the workflow from the list (source **Database**). Its inputs, `title` and `message`, appear automatically.

3 Fill in the inputs with expressions, such as `New order` and `{{ $json.customer }}` spent `{{ $json.total }}`.

4 Choose the mode.

Run once with all items sends all items in one call; **Run once for each item** calls it for every item.

5 Run it.

Open the node afterwards and click **View sub-execution** to jump into the sub-workflow's run.

14.4 How data passes between workflows

1. The Execute Sub-workflow node sends its data to the sub-workflow's trigger.
2. The sub-workflow runs.
3. Whatever its **last node** outputs comes back to the Execute Sub-workflow node in the parent, and the parent carries on.

If you don't need the result, turn off the option **Wait for Sub-Workflow Completion** so the parent continues immediately, like posting a letter rather than waiting for a reply.

14.5 Convert part of a workflow

Already have a long workflow? n8n can convert a selection of connected nodes into a sub-workflow: it moves them into a new workflow and puts an Execute Sub-workflow node in their place. Search the docs for "Convert to sub-workflows" for the current steps. Check the new sub-workflow's inputs afterwards.

14.6 Ideas for your library

- Notify me (Slack, email or Telegram)
 - Look up or create a CRM contact
 - Log an event to a sheet or data table
 - Summarise text with AI
 - Format a date or currency your way
 - Get an access token for an awkward API
-
- ☐ I've built a sub-workflow with defined inputs
 - ☐ I've called it with Execute Sub-workflow and viewed the sub-execution
 - ☐ I know the last node's output is returned to the parent
 - ☐ I know sub-workflow runs don't count against my execution quota

Data Tables: A Spreadsheet Inside n8n

Workflows often need to remember things between runs: which emails were already handled, a list of VIP customers, the prompt text your AI uses. You could use Google Sheets or a database, but n8n has **data tables** built in: simple tables that live inside your n8n project.

IN THIS CHAPTER

- What data tables are good (and not good) for
- Creating a table by hand or from a CSV
- Reading and writing rows with the Data Table node
- A worked example: never process the same item twice
- Limits and alternatives

15.1 Good uses

- Remembering what you've already processed, to avoid duplicates
- Lookup tables, such as product codes to names
- Reusable prompts and message templates
- A simple log of what your workflows did
- Test cases for checking AI workflows

15.2 Create a table

MAKE A "PROCESSED" TABLE

- 1 **Open your project** (or Overview) and select the **Data tables** tab.

- 2 **Select Create Data table** from the button in the top right.
- 3 **Name it** “processed_items”.
- 4 **Choose From scratch**, then add columns: `item_id` (string) and `processed_at` (date).

You can also choose **Import CSV** to build a table from a spreadsheet file, and later **Download CSV** from the table’s menu. In the table view you can add, edit, reorder and delete rows and columns by hand.

15.3 The Data Table node

In a workflow, the **Data Table** node works with two resources:

RESOURCE	OPERATIONS
Row	Insert, Get, Update, Upsert, Delete, If Row Exists, If Row Does Not Exist
Data table	Create, List, Update, Delete

Upsert means “update the row if it exists, otherwise insert it.” **If Row Does Not Exist** passes through only the items that aren’t in the table yet, which makes it a one-node duplicate filter.

15.4 Worked example: skip what you’ve seen

PROCESS EACH ITEM ONLY ONCE

- 1 **Start with your trigger and data**, for example an RSS Read node returning posts with a `link` field.
- 2 **Add a Data Table node**, resource **Row**, operation **If Row Does Not Exist**. Choose `processed_items` and match `item_id` against `{{ $json.link }}`.
- 3 **Do the work** on the items that come out: send them to Slack, summarise them, and so on.
- 4 **Record them.**
Add another Data Table node, operation **Insert**, setting `item_id` to `{{ $json.link }}` and `processed_at` to `{{ $now.toISO() }}`.

Run it twice: the second run finds nothing new and stops early.

15.5 Limits

- By default, all data tables in an instance share **200 MiB** of storage. n8n warns at 80%. Past the limit, inserts and updates fail. Self-hosted owners can raise it with the `N8N_DATA_TABLES_MAX_SIZE_BYTES` environment variable.
- Tables belong to a project, and everyone in the project can use them.
- The Code node can't read data tables directly; use the Data Table node and pass the results in.

15.6 Data tables, variables or a real database?

NEED	USE
Rows and columns, moderate size	Data tables
A few constant values used everywhere (paid plans)	Custom variables (<code>\$vars</code>)
Large data, many users, complex queries	PostgreSQL, Airtable, Google Sheets or similar

- ☐ I've created a data table by hand
- ☐ I've used Insert and If Row Does Not Exist in a workflow
- ☐ I know the 200 MiB default limit
- ☐ I know when to move to a real database

Saving, Publishing and Versions

n8n separates *working on* a workflow from *running it for real*. Understanding this saves you from the two classic mistakes: “I changed it but nothing happened” and “I broke the live version while experimenting.”

IN THIS CHAPTER

- Autosave and drafts
- Publishing, unpublishing and the publish button's states
- Workflow history: viewing and restoring old versions
- Naming versions
- Exporting, importing and sharing workflows as files
- Tags, folders and settings that help you stay organised

16.1 Autosave and drafts

Changes save automatically as you edit, usually within a few seconds; there's no save button to remember. Every edit stays in **draft** until you publish. Only one person can edit a workflow at a time: others see it read-only until the editor becomes inactive.

16.2 Publishing

Publishing makes the current version live. Production executions then always use that published version, not your latest edits. When you publish:

- Webhook and Form triggers start answering on their production URLs.
- Schedules start running.
- App events start triggering the workflow.

PUBLISH A WORKFLOW

- 1 Click **Publish** in the top bar, or press **Shift + P**.
- 2 Optionally name the **version** and add a description of what changed.
- 3 Click **Publish** in the dialog. The button shows **Publishing** while n8n registers your triggers in the background.

The **Publish** button tells you the state at a glance:

BUTTON STATE	MEANING
Disabled	Nothing to publish
Active	Unpublished changes are waiting
Published	Live and up to date
Published, has changes	Live, but your newer edits aren't live yet
Published, partial	Some triggers failed to activate; hover to see which, then publish again
Failed to publish	No trigger activated; fix the problem and retry

To **unpublish** (stop a workflow from running), use the arrow next to **Publish**, the workflow list, or version history. Shortcut: **Ctrl / Cmd + U**.

TIP

Because edits stay in draft, you can safely improve a live workflow during the day and publish when you're happy. Test with **Execute workflow** first.

16.3 Workflow history

Each autosave creates a new **version**. Open history with the clock icon in the top bar. For each version you can:

- **Restore version:** make it your current draft (it doesn't change production until you publish);
- **Clone to new workflow:** start a copy from it;
- **Open version in new tab:** compare side by side;
- **Download:** save it as JSON;
- **Name version:** protect it from automatic cleanup.

How far back you can go depends on your plan: the last 24 hours for everyone, five days on Cloud Pro, and full history on Enterprise. Named versions are available on Cloud Pro and Enterprise and self-hosted Business and Enterprise, and are never pruned.

16.4 Exporting and importing

Workflows are JSON files, which makes them easy to back up and share.

- **Download:** workflow menu (...) › **Download**.
- **Import:** in a new workflow, use the workflow menu › **Import from File** or **Import from URL**, or simply **paste** the JSON onto the canvas.
- **Copy nodes:** select nodes, `Ctrl / Cmd + C`, and paste into another workflow or a message.

⚠ WARNING

Exported workflows don't contain credential secrets, but they do contain node settings, which can include things like email addresses, sheet IDs or URLs. Check before posting a workflow publicly, and remove anything personal.

16.5 Staying organised

- **Names:** “Weekly sales report (Mon 8am)” beats “My workflow 7.”
- **Tags:** add from the top bar, then filter the list by tag.
- **Folders and projects:** group related workflows.
- **Descriptions and sticky notes:** explain the why, not just the what.
- **Archive** workflows you no longer use rather than deleting them straight away.

16.6 Workflow settings worth knowing

Open the workflow menu (...) › **Settings**:

SETTING	USE IT TO
Error Workflow	Run an alert workflow when this one fails (Chapter 17)
This workflow can be called by	Control which workflows may use it as a sub-workflow
Timezone	Override the instance time zone for schedules and dates
Save failed / successful production executions	Choose what to keep in the Executions list
Save manual executions	Keep or discard your test runs
Timeout Workflow	Stop runs that take too long

16.7 Git and environments (advanced)

On self-hosted Business and Enterprise plans, **source control** connects n8n to a Git repository so you can keep separate development and production instances and move changes between them in a controlled way. Most individuals don't need this: version history and exported JSON files are plenty.

- ☐ I know edits stay in draft until I publish
- ☐ I can read the Publish button's state
- ☐ I've restored or cloned an older version
- ☐ I've exported a workflow as JSON and imported it again
- ☐ I've set a time zone and error workflow in workflow settings

Handling Errors Gracefully

Things go wrong: an API is down for a minute, a customer leaves a field blank, a password expires. A reliable workflow expects this. This chapter shows the tools n8n gives you to retry, route around, and get told about problems before anyone else notices.

IN THIS CHAPTER

- Retry On Fail for flaky services
- On Error: stop, continue, or take an error branch
- Error workflows and the Error Trigger
- Failing on purpose with Stop and Error
- Patterns for missing data and rate limits

TOOL	WHAT IT DOES
Retry On Fail (node settings)	Automatically retry a flaky step, e.g. 3 times, waiting 5 seconds
On Error > Continue (node settings)	Keep the workflow going and send the error down a separate output
Error workflow	In workflow Settings , choose a workflow that starts with an Error Trigger ; it runs whenever this one fails, so you can email or Slack yourself the details
Stop and Error node	Deliberately fail with a clear message when something's wrong

17.1 Retry On Fail

Open a node, switch to its **Settings** tab, and turn on **Retry On Fail**. Set **Max Tries** (for example 3) and **Wait Between Tries** (for example 5,000 ms). Use it on anything that talks over the internet: HTTP requests, AI models, app nodes. Most temporary failures disappear on the second try.

17.2 On Error

Also in a node's **Settings**, **On Error** decides what happens when the node fails after any retries:

CHOICE	RESULT
Stop Workflow (default)	The run fails; the error workflow fires
Continue	Carry on with the error details as the output
Continue (using error output)	The node gets a second, red error output you can connect to fallback steps

The error output is the most useful: for example, if enriching a lead fails, send it down the error output to a “needs manual review” sheet instead of losing it.

17.3 Error workflows

CREATE AN “ALERT ME” ERROR WORKFLOW

- 1 **Create a new workflow** called “Error handler”.
- 2 **Add an Error Trigger** as the first node.
- 3 **Add a message node**, for example Gmail or Slack, with a message like:

TEXT

```

⚠ {{ $json.workflow.name }} failed
Node: {{ $json.execution.lastNodeExecuted }}
Error: {{ $json.execution.error.message }}
Details: {{ $json.execution.url }}

```

- 4 **Save it.**
An error workflow doesn't need publishing to work.
- 5 **Attach it.**
In each important workflow, open the workflow menu (...) > **Settings**, choose “Error handler” under **Error Workflow**, and save.

The Error Trigger receives the failed workflow's name and ID, the execution ID and a link to it, the error message, and the last node that ran. If the failure happened in the trigger itself (for example a schedule couldn't activate), the data arrives under `trigger` instead, with less detail.

TIP

Make one “Alert me” error workflow, with Error Trigger → Gmail or Slack, and set it as the error workflow for everything you publish. You'll know about problems before anyone else does.

17.4 Stop and Error

Sometimes nothing technically fails, but the data is wrong: an order with a negative total, a form with no email. Add an If node to check, and on the bad path add **Stop and Error** with a clear message such as `Order {{ $json.id }} has no email`. The run fails cleanly and your error workflow tells you why.

17.5 Patterns for common problems

PROBLEM	PATTERN
A field is sometimes missing	Use <code>{{ \$ifEmpty(\$json.phone, 'none') }}</code> , or an If node checking is empty
The API rate-limits you (error 429)	HTTP Request Batching , or Loop Over Items with a Wait node
A service is down for hours	Error output → store the item in a data table → a scheduled workflow retries later
A node returns no items and the flow stops	Turn on Always Output Data in node settings, then check for emptiness
Long runs time out	Split into sub-workflows; set Timeout Workflow deliberately

NOTE

When a node outputs zero items, the nodes after it don't run. That's usually what you want (“nothing new, nothing to do”), but it surprises people. **Always Output Data** makes the node output one empty item instead.

- ☐ Retry On Fail is on for my internet-facing nodes
- ☐ I've used an error output for at least one fallback path
- ☐ I have an “Error handler” workflow
- ☐ Every published workflow points at it
- ☐ I validate key data and fail clearly with Stop and Error

Debugging Executions

When a workflow misbehaves, n8n has kept a detailed record of exactly what happened. Learning to read that record turns debugging from guesswork into a calm, methodical process.

IN THIS CHAPTER

- The Executions list and what to look for
- Reading a failed execution node by node
- Debug in editor and Copy to editor
- Retrying failed runs
- Custom execution data for searching
- A step-by-step debugging routine

18.1 The Executions list

Click the **Executions** tab of a workflow to see every saved run, or open **Overview › Executions** to see runs of all workflows you can access. Each row shows the status (**Succeeded**, **Error**, **Running**, **Waiting**, **Queued**), start time and duration. Filter by status or date to find the one you need.

NOTE

Which runs appear depends on workflow settings (**Save failed / successful production executions**, **Save manual executions**) and on your plan's retention limits. On Cloud Starter, executions are kept for up to 7 days; on Pro, up to 30.

18.2 Reading a failed execution

Click a failed execution. The canvas opens read-only with the run's data. Nodes that ran show a green tick, and the failing node is red. Click any node to see its input and output exactly as they were. Open the red node to read the error message: it usually says what went wrong, and often which field.

18.3 Debug in editor

For a failed run, select **Debug in editor**; for a successful one, **Copy to editor**. n8n loads that run's data into your workflow and **pins** it on the first node. Now you can change the broken node, click **Execute workflow**, and re-run with the exact data that failed, without waiting for the real event to happen again. Remember to unpin afterwards.

NOTE

Debug in editor is available on all Cloud plans and on self-hosted Registered Community, Business and Enterprise. Self-hosted users get it by registering for free.

18.4 Retrying

From the Executions list you can **retry** a failed execution, either with the currently published workflow (after you've fixed it) or with the original version. Retries are a quick way to recover the items that failed during an outage.

18.5 Custom execution data

You can tag executions with your own searchable data, such as an order number, then filter the Executions list by it. Use the **Execution Data** node, or in a Code node:

JAVASCRIPT

```
$execution.customData.set('orderId', $json.id);
```

Custom execution data is available on Cloud Pro and Enterprise and self-hosted Registered Community and Enterprise.

18.6 A debugging routine

WHEN SOMETHING'S WRONG

- 1 **Find the run.**
Open Executions and click the failed or suspicious run.
- 2 **Find the first surprise.**
Walk through nodes left to right. Check the item count and values in each output. The bug is usually at the first node whose output isn't what you expected, not at the red one.
- 3 **Read the error carefully.**
Copy it into a search of the community forum if it's unclear.
- 4 **Load the data.**
Use **Debug in editor** so you can experiment with real input.
- 5 **Fix one thing, then test.**
Execute the step, check output, repeat.
- 6 **Publish the fix**, then retry the failed executions if needed.

PROMPT

My n8n workflow fails at the node "[node name]" with this error:
[paste the error]. The node's input looks like this: [paste the input JSON].
The node is configured like this: [describe the settings]. Explain in plain English what's wrong and how to fix it.

18.7 Decoding common error types

ERROR SAYS	USUALLY MEANS
401 Unauthorized	Credential wrong or expired: reconnect it
403 Forbidden	The account lacks permission for this action
404 Not Found	Wrong URL, ID or resource
429 Too Many Requests	Slow down: batching, Wait, Retry On Fail
500, 502, 503	The other service had a problem: retry later
"Referenced node is unexecuted"	An expression points at a node that didn't run on this path
"JSON parameter needs to be valid JSON"	A body or field contains broken JSON: check quotes and commas
Workflow timed out	Took longer than the allowed time: split it up

- ☐ I can find a run in the Executions list
- ☐ I can find the first node whose output looks wrong
- ☐ I've used Debug in editor to re-run with real data
- ☐ I know how to retry failed executions

AI in n8n: The Basics

n8n has three layers of AI: AI *steps* inside workflows, AI *agents* that decide what to do, and AI that *builds workflows for you*. This chapter starts with the simplest and most reliable: using a language model as one step in an ordinary workflow.

IN THIS CHAPTER

- Language models, prompts and tokens in plain English
- Connecting a chat model: your own key or Gateway credits
- The ready-made AI nodes: chain, extractor, classifier, sentiment, summariser
- Getting structured answers you can use in later nodes
- n8n Assistant and other ways AI can build for you

19.1 A few AI words

WORD	MEANING
LLM (large language model)	The AI that reads and writes text, such as Claude, GPT or Gemini
Prompt	The instructions and text you send the model
System message	Standing instructions about role and style
Token	A chunk of text (about ¾ of a word); providers charge per token
Chat model	The n8n sub-node that connects to an LLM provider
Hallucination	When a model confidently states something false

19.2 AI steps in a workflow

Add an AI model node anywhere to summarize, classify, translate, or extract information. The **Basic LLM Chain**, **Information Extractor**, **Text Classifier**, and **Sentiment Analysis** nodes cover the common jobs. Connect a **chat model** (OpenAI, Anthropic Claude, Google Gemini, Mistral, Ollama for local models, and more) using your API key.

NODE	DOES	EXAMPLE
Basic LLM Chain	Sends a prompt, returns the answer	"Summarise this article in three bullets"
Information Extractor	Pulls named fields out of text	Invoice number, date and total from an email
Text Classifier	Sorts text into categories you define	Billing / technical / sales / spam
Sentiment Analysis	Judges tone	Positive, neutral or negative review
Summarization Chain	Summarises long documents in chunks	A 40-page PDF

TIP

On n8n Cloud Starter and Pro plans you can choose **Use Gateway credits** in a model's credential field. n8n handles billing, so you don't need your own OpenAI or Anthropic account. Great for getting started.

19.3 Connecting a chat model

SUMMARISE TEXT WITH CLAUDE

- 1 Add a Basic LLM Chain** after a node that outputs text (say, an RSS item with `content`).
- 2 Set the prompt.**
Choose **Define below** and write: `Summarise this in two sentences for a busy reader: {{ $json.content }}`.
- 3 Attach a chat model.**
Click the **Model** connector under the node and pick **Anthropic Chat Model** (or OpenAI, Gemini...).
- 4 Add a credential.**
Paste an API key from the provider's console, or choose **Use Gateway credits** on Cloud.
- 5 Pick a model** from the list, then **Execute step**. The answer appears in the output, ready for Slack or email.

NOTE

Gateway credits are available on Cloud Starter and Pro (not Enterprise or self-hosted). New accounts and trials include a small free amount. Top-ups expire 12 months after purchase. Rates per model are on n8n's service pricing page.

19.4 Getting structured answers

A paragraph of text is hard for later nodes to use. You want fields: `category`, `priority`, `reply`. Two ways:

- **Information Extractor:** list the attributes you want with a short description of each, or give a JSON example. The output has one field per attribute.
- **Require Specific Output Format** on a Basic LLM Chain or AI Agent, with a **Structured Output Parser** attached. Give it an example like the one below and n8n makes the model fill in exactly that shape.

EXAMPLE FOR THE STRUCTURED OUTPUT PARSER

```
{
  "category": "billing",
  "priority": "high",
  "summary": "Customer was charged twice in October",
  "suggested_reply": "Hi Sam, sorry about that..."
}
```

19.5 Writing good prompts

- ☐ Say who the AI is and who the output is for (“You are a support assistant replying to customers”)
- ☐ Give the exact task and format (“Reply in under 80 words, plain text, no greeting line”)
- ☐ Include the data clearly, between markers like `---`
- ☐ Give one example of a perfect answer
- ☐ Say what to do when unsure (“If the category is unclear, answer `other`”)

19.6 Running models locally

Self-hosted users can run open models on their own hardware with **Ollama** and connect them with the **Ollama Chat Model** node. Your data never leaves your server, and there's no per-token bill, but you need a capable machine, and small models are less accurate than the big cloud ones.

19.7 Let AI build workflows for you

- **n8n Assistant** (in preview): a chat panel inside n8n that can plan, build, test and fix workflows from a description, and help with tasks like publishing. It's available on Cloud Starter and Pro and on self-hosted Community and Business. It uses **Assistant credits**: a monthly allowance comes with Cloud plans (1,600 a month on Starter). On self-hosted, you add your own model key.
- **From Claude or another MCP client**: connect to n8n's instance-level MCP server and say "build a workflow that saves Gmail attachments to Google Drive" (Chapter 22).

⚠ WARNING

Review AI-built workflows before publishing, especially nodes that send messages, delete data, or spend money. Test with pinned or sample data first.

PROMPT

Build an n8n workflow that runs every weekday at 8am, reads new rows from my Google Sheet "Leads" added in the last 24 hours, writes a two-line summary of each with AI, and posts the summaries to the Slack channel #sales. Use Gateway credits for the model. Don't publish it; I'll review it.

- ☐ I've connected a chat model with a key or Gateway credits
- ☐ I've used Basic LLM Chain for a summary
- ☐ I've extracted structured fields with Information Extractor or an output parser
- ☐ I review AI-built workflows before publishing

The AI Agent Node in Depth

An AI step does one fixed job. An **AI agent** decides *what to do*: it reads a request, chooses which tools to use, uses them, looks at the results, and keeps going until it has an answer. This chapter explains how agents work in n8n and how to make them dependable.

IN THIS CHAPTER

- How an agent thinks: the reason-act loop
- The parts: model, memory, tools, system message
- Building a chatbot agent step by step
- Memory types and sessions
- Tools: app nodes, HTTP, sub-workflows, `$fromAI()`
- Human approval, guardrails and limits
- n8n's first-class Agents (preview)

20.1 How an agent works

An agent runs a loop. It **thinks** (“to answer this I need today’s calendar”), **acts** (calls the calendar tool), **observes** the result, and repeats until it can reply. Each tool has a name and a description; the model reads those descriptions to decide what to use, so clear descriptions matter enormously.

20.2 The AI Agent node

The **AI Agent** node is a mini-agent inside a workflow. You attach:

PART	WHAT IT'S FOR
Chat model	The AI brain (e.g. Claude or GPT)
Memory	Remembers the conversation, e.g. Simple Memory , or a database for persistence
Tools	Things the agent can do: Gmail, Google Calendar, HTTP Request, a sub-workflow, or an MCP Client Tool
System message	Instructions: who it is and how to behave

The agent needs at least one tool. Its prompt comes either **from the previous node automatically** (it expects a field called `chatInput`, which the Chat Trigger provides) or from a prompt you **define below**.

BUILD A SIMPLE CHATBOT AGENT

- 1 **Add a Chat Trigger** as the first node.
- 2 **Add an AI Agent node** after it.
- 3 **Attach a chat model** (click **Chat Model +**) and choose, for example, **Anthropic Chat Model** with your credential or Gateway credits.
- 4 **Attach Simple Memory** so it remembers the conversation.
- 5 **Attach a tool**, for example **Google Calendar** with the operation *Get many events*, so it can answer “what’s on my schedule?”
- 6 **Write the system message:**
“You are my friendly scheduling assistant. Use the calendar tool to answer questions about my day. Be brief.”
- 7 **Click Open chat** at the bottom of the canvas and talk to it.

20.3 Writing the system message

The system message is the agent's job description. A good one covers:

PROMPT

You are the support assistant for Bright Bikes, a bicycle shop.
 Goal: answer customer questions about orders, opening hours and repairs.
 Tools: use "Find order" to look up orders by email. Use "Knowledge base" for policies. Never guess an order status; always look it up.
 Style: friendly, plain English, under 100 words.
 If you can't help, say so and offer to pass the message to a human.
 Today's date is {{ \$now.toFormat('d LLLL yyyy') }}.

TIP

Models don't know today's date. Adding it to the system message with an expression prevents a whole class of mistakes.

20.4 Agent options

OPTION	USE IT TO
Max Iterations	Cap how many think-act rounds it can take (default 10)
Return Intermediate Steps	Include the tool calls in the output, for checking its reasoning
Require Specific Output Format	Attach an output parser for structured results
Enable Streaming	Show the reply word by word in chat
Automatically Passthrough Binary Images	Let the model see images from earlier nodes

20.5 Memory

Without memory, every message is a fresh start. Memory sub-nodes store the conversation:

MEMORY NODE	STORES HISTORY	GOOD FOR
Simple Memory	Inside n8n	Getting started, single-server setups
Postgres Chat Memory	In a PostgreSQL database	Production, survives restarts
Redis Chat Memory	In Redis	Fast, production
MongoDB Chat Memory	In MongoDB	If you already use MongoDB

Two settings matter: the **Session Key**, which keeps different users' conversations apart (the Chat Trigger supplies a session ID automatically), and the **Context Window Length**, how many past exchanges to send back to the model. Longer memory means better context but more tokens.

 WARNING

Simple Memory doesn't work for published workflows when n8n runs in queue mode (Chapter 26). Use Postgres or Redis chat memory there.

20.6 Tools

Almost any app node can be attached as a tool: Gmail, Sheets, Notion, Slack, Calendar, HTTP Request and hundreds more. Other useful tools include **Call n8n Workflow Tool** (run a sub-workflow), **Code Tool**, **Calculator**, **Think** (gives the model room to plan), **Wikipedia**, **SerpAPI** or **SearXNG** for web search, **Vector Store Question Answer Tool** (Chapter 21), and **MCP Client Tool** (Chapter 22).

When you attach an app node as a tool, you choose which parameters you fix and which the AI fills in. For any field you want the model to decide, let the AI define it (the field shows an option for this); n8n writes it as a `$fromAI()` expression, which you can also type yourself:

TEXT

```
{{ $fromAI('recipient_email', 'The email address of the customer', 'string') }}
```

Fix everything you can (the sender, the channel, the sheet) and leave open only what the agent truly needs to choose. Fewer open fields means fewer mistakes.

 TIP

Give each tool a clear name and description: “Find order: looks up a customer's orders by email address. Returns order number, status and date.” The model chooses tools by reading these.

20.7 Human review for risky tools

You can make the agent ask a person before it uses certain tools. Click the agent's tool connector, find **Human review** in the tools panel, choose a channel (chat, Slack, Telegram and others), and connect the sensitive tools to it. When the agent wants to use one, the workflow pauses and sends an approval request; the action runs only if someone approves.

20.8 Guardrails

The **Guardrails** node checks text before it reaches your model, or before a model's answer reaches a customer. **Check Text for Violations** sends failing items to a **Fail** branch; **Sanitize Text** replaces sensitive content instead. Available checks include keywords, jailbreak attempts, NSFW content, personal data (PII) such as card or phone numbers, secret keys, staying on topic, URLs, and your own custom or regex rules. LLM-based checks need a chat model attached.

20.9 Keeping agents dependable

- ☐ One clear job per agent; split big jobs into several agents or sub-workflows
- ☐ Fixed parameters wherever possible, `$fromAI()` only where needed
- ☐ Human review on anything that sends, deletes or pays
- ☐ Guardrails on public-facing input and output
- ☐ Max Iterations set to a sensible number
- ☐ Tested with tricky questions before publishing

20.10 Agents (first-class, in preview)

Newer versions of n8n also offer **Agents** as their own kind of project item, alongside workflows. You configure a model, instructions, tools, skills, and a knowledge base in the **Agent Builder**, then let people reach the agent through chat, channels, or schedules. Agents can call your workflows as tools. The feature is in preview, so expect changes. Owners can switch it on or off in **Settings** > **Agents**.

CHOOSE	WHEN
AI Agent node in a workflow	The flow is mostly fixed, with AI as one step
A first-class Agent	The AI owns the conversation and decides what to do

Knowledge for Your AI: Vector Stores and RAG

A language model knows a lot about the world, but nothing about *your* price list, handbook or support articles. **RAG** (retrieval-augmented generation) fixes that: before answering, the AI looks up the most relevant passages from your documents and uses them. This chapter explains the idea and builds a working knowledge base.

IN THIS CHAPTER

- Embeddings and vector stores, without the maths
- The two halves of RAG: loading and asking
- Building a knowledge base from documents
- Connecting it to an agent as a tool
- Choosing a vector store for production
- Making answers more accurate

21.1 The idea in plain English

Imagine a librarian who has read every page of your documents and sorted them by *meaning*. When a question arrives, they pull out the three most relevant pages and hand them to the writer, who answers using only those pages.

- An **embedding** is a list of numbers that captures the meaning of a piece of text. Texts with similar meanings get similar numbers. An **embeddings model** (for example OpenAI, Gemini, Cohere, Mistral or Ollama embeddings) creates them.
- A **vector store** is a database that stores those numbers alongside the text, and can quickly find the passages closest in meaning to a question.
- **Chunking** means cutting long documents into passages (a few paragraphs each) so each one can be found separately. A **text splitter** does this.

21.2 The two halves of RAG

HALF	RUNS	STEPS
Loading (ingestion)	When documents change	Get files → split into chunks → embed → store
Asking (retrieval)	Every question	Embed the question → find similar chunks → answer with them

21.3 Build a knowledge base

LOAD DOCUMENTS INTO A VECTOR STORE

- 1 Start with a trigger and your documents.**
For a first test, use a Manual Trigger and a Google Drive node that downloads a few PDFs or Google Docs.
- 2 Add a vector store node** and choose the **Insert Documents** operation. To begin, use **Simple Vector Store** and give it a **Memory Key** such as `handbook`.
- 3 Attach an embeddings model** to its **Embedding** connector, for example **Embeddings OpenAI**.
- 4 Attach a document loader**
Default Data Loader, set to binary (for files) or JSON (for text fields).
- 5 Attach a text splitter** to the loader: **Recursive Character Text Splitter** is a good default. Try a chunk size of about 1,000 characters with 200 characters of overlap.
- 6 Execute the workflow.**
The output shows how many chunks were stored.

21.4 Ask questions with an agent

CONNECT THE KNOWLEDGE BASE TO A CHATBOT

- 1 **Build the chatbot** from Chapter 20: Chat Trigger → AI Agent with a chat model and memory.
- 2 **Add a vector store as a tool.**
On the agent's **Tool** connector, add the same vector store node with the operation **Retrieve Documents (As Tool for AI Agent)**, and the same Memory Key, `handbook`.
- 3 **Describe the tool.**
Name it `company_handbook` and describe it: "Searches the staff handbook. Use it for any question about policies, holidays or expenses."
- 4 **Use the same embeddings model** as when loading. Different models produce incompatible numbers.
- 5 **Set a Limit** of 4 to 6 chunks, so the model gets enough context without drowning.
- 6 **Tell the agent to rely on it.**
Add to the system message: "Answer only from the handbook tool. If it doesn't contain the answer, say you don't know."

21.5 Choosing a vector store

⚠ WARNING

Simple Vector Store keeps data in n8n's memory. It's lost when n8n restarts, and every user on the instance can access it. Use it for experiments only.

VECTOR STORE	GOOD FOR
PGVector	You already run PostgreSQL (add the pgvector extension)
Supabase	Hosted Postgres with a free tier
Qdrant	Fast, open source, self-hostable or cloud
Pinecone	Fully managed cloud service
Others	Chroma, Milvus, MongoDB Atlas, Redis, Weaviate, Zep, Azure AI Search

The workflow shape is the same whichever you pick: only the node and credential change.

21.6 Keeping it fresh

Run the loading workflow whenever documents change: on a schedule, or with a Google Drive trigger on file updates. To avoid duplicates, clear the old chunks for a document before inserting new ones (Simple Vector Store has a **Clear Store** option; other stores let you delete by metadata or use a separate collection per source).

21.7 Improving answers

PROBLEM	TRY
Answers miss obvious facts	Smaller chunks, more results (higher Limit)
Answers mix up topics	Larger chunks with overlap; clearer tool description
Agent ignores the knowledge base	Say in the system message to always search it first
Made-up answers	"If it isn't in the documents, say you don't know"
Slow or expensive	Fewer chunks; a cheaper model for simple questions
Ranking is poor	Add a reranker such as Reranker Cohere

- ☐ I can explain embeddings and vector stores in my own words
- ☐ I've loaded documents with an embeddings model, loader and splitter
- ☐ My agent uses the vector store as a tool with a clear description
- ☐ I use the same embeddings model for loading and asking
- ☐ I've picked a persistent vector store for production

MCP: n8n and Other AI Apps

MCP (Model Context Protocol) is the standard way for AI apps to use outside tools, like a universal plug socket. n8n speaks MCP in both directions: your agents can use other MCP servers, and AI apps such as Claude can use n8n. See the *MCP* manual in this series for the bigger picture.

IN THIS CHAPTER

- The three ways n8n uses MCP
- Giving your agent tools from MCP servers
- Turning a workflow into an MCP server
- Letting Claude or ChatGPT search, run and build your workflows

FEATURE	DIRECTION	USE IT TO
MCP Client Tool node	n8n uses other MCP servers	Give your AI Agent tools from GitHub, Notion, Linear, and more. The node panel also has a one-click MCP Servers registry
MCP Server Trigger node	Other AI apps use one n8n workflow	Turn a workflow into a custom MCP server that Claude or Cursor can call
Instance-level MCP server	Other AI apps use all of n8n	Let Claude, ChatGPT, or a coding agent search, run, <i>and build</i> your workflows from a chat

22.1 Using MCP servers in your agents

ADD A REGISTRY MCP SERVER IN ONE CLICK

- 1 Open your AI Agent's tools panel by clicking **Tool +**.
- 2 Search for the **service** (for example Notion or Linear) or open the **MCP Servers** section.
- 3 Select it and sign in when asked. n8n creates the MCP credential for you.

4 Choose Tools to include

all, a selected list, or all except some.

If a server isn't in the registry, add an **MCP Client Tool** and enter its **endpoint URL** (for example `https://mcp.notion.com/mcp`) and authentication (bearer token, header, multiple headers or OAuth2). To use an MCP tool as an ordinary workflow step rather than through an agent, use the **MCP Client** node and pick the tool from its list.

USE A BUILT-IN TOOL WHEN...	USE AN MCP SERVER WHEN...
You know the exact action needed	The agent needs a range of actions
You want to fix most parameters	You're happy for it to choose tools and arguments

22.2 Turn a workflow into an MCP server

EXPOSE TOOLS WITH THE MCP SERVER TRIGGER

- 1 Create a workflow with an **MCP Server Trigger**.
- 2 Attach tools to it: app nodes, or **Call n8n Workflow Tool** nodes that run your own workflows.
- 3 Set Authentication to **Bearer auth** or **Header auth**, and choose a memorable **Path**.
- 4 Publish, then copy the **Production URL** from the top of the node.
- 5 Add it to your AI app as a remote MCP server, with the token.

For apps that only support local (stdio) servers, such as some Claude Desktop setups, n8n's docs show bridging with `mcp-remote`:

CLAUDE_DESKTOP_CONFIG.JSON

```
{
  "mcpServers": {
    "n8n": {
      "command": "npx",
      "args": ["mcp-remote", "<MCP_URL>",
        "--header", "Authorization: Bearer ${AUTH_TOKEN}"],
      "env": { "AUTH_TOKEN": "<MCP_BEARER_TOKEN>" }
    }
  }
}
```

22.3 Instance-level MCP: let AI build for you

The instance-level server connects an AI app to your whole n8n. From a chat, the AI can search workflows, run the ones you've enabled, create and edit workflows (n8n 2.13 onward), and manage data tables.

CONNECT CLAUDE TO YOUR N8N

1 Enable it.

As owner or admin, open **Settings** > **Instance-level MCP** and select **Enable MCP access**.

2 Connect a client.

In **Connection details**, select **Connect**. Choose **OAuth (recommended)** or an API key. Web clients offer one-click setup; otherwise copy the **Server URL**, which ends in `/mcp-server/http`.

3 Approve access when the client sends you back to n8n.

4 Enable workflows.

Clients can only see previews until you enable a workflow: open it, go to the workflow menu > **Settings** and toggle **Available in MCP**, or use **Workflows enabled** on the MCP settings page.

☆ IMPORTANT

Every connected client can see every workflow you've enabled for MCP. Enable only what you're happy for AI apps to run. You can revoke a client at any time under **Connected clients**.

⚠ WARNING

Review AI-built workflows before publishing, especially nodes that send messages, delete data, or spend money. Test with pinned or sample data first.

PROMPT

Using my n8n MCP connection, build a workflow that triggers when a file is added to my Google Drive folder "Receipts", extracts the vendor, date and total with AI, and appends them to the Google Sheet "Expenses 2026". Test it with a sample, show me the result, and don't publish it.

- ☐ I've given an agent tools from an MCP server
- ☐ I've exposed one workflow with the MCP Server Trigger and authentication
- ☐ I've connected an AI app to instance-level MCP

Recipes: Everyday Automations

The next three chapters are a cookbook: twelve complete workflows you can build in an afternoon each. Every recipe lists what you need, the steps, and ideas to extend it. Build them as written first, then make them yours. Exact field names in app nodes can vary slightly between versions; pick the closest match.

IN THIS CHAPTER

- Recipe 1: RSS-to-Slack daily digest
- Recipe 2: a daily report email
- Recipe 3: nightly file backup
- Recipe 4: social post scheduler

TIP

For every recipe: test each node as you add it, pin data on slow or expensive steps, attach your error workflow (Chapter 17), and only then publish.

23.1 Recipe 1: RSS-to-Slack digest

Goal: every morning, post a short AI-written digest of new articles from your favourite sites to a Slack channel.

You need: Slack credential, a chat model (or Gateway credits). **Nodes:** Schedule Trigger, RSS Read, Remove Duplicates, Limit, Aggregate, Basic LLM Chain, Slack.

BUILD THE DIGEST

- 1 Schedule Trigger:**
Days, at 8am.
- 2 RSS Read:**
paste a feed URL (most blogs and news sites have one; try adding `/feed` or `/rss` to the site address). For several feeds, put the URLs in an Edit Fields node as a list, Split Out, and feed each into RSS Read using `{{ $json.url }}`.
- 3 Remove Duplicates:**
operation **Remove Items Processed in Previous Executions**, deduping on `{{ $json.link }}`. Now you only see new articles.
- 4 Limit:**
keep 10 items so the digest stays short.
- 5 Aggregate:**
combine the `title`, `link` and `contentSnippet` fields into lists, making one item.
- 6 Basic LLM Chain:**
prompt: `Write a friendly Slack digest. For each article give the title as a link and one sentence on why it matters. Articles: {{ JSON.stringify($json) }}`.
- 7 Slack:**
send a message to `#reading` with the chain's `text` output.

Extend it: add a Text Classifier to tag articles by topic; post only on weekdays; send to email or Telegram instead.

23.2 Recipe 2: daily report email

Goal: at 6pm, email yourself a summary of today's sales from a Google Sheet.

Nodes: Schedule Trigger, Google Sheets, Filter, Summarize, Code (or Edit Fields), Gmail.

BUILD THE REPORT

- 1 Schedule Trigger:**
Days, at 6pm.
- 2 Google Sheets:**
operation to get rows from your "Sales" sheet.

- 3 Filter:**
keep rows where `{{ $json.date }}` equals `{{ $today.toFormat('yyyy-MM-dd') }}` (match your sheet's date format).
- 4 Summarize:**
sum `amount`, count rows, grouped by `product` if you like.
- 5 Code:**
build an HTML table with recipe 5 from Chapter 10, adding a total line.
- 6 Gmail:**
send to yourself, subject `Sales {{ $today.toFormat('d LLL') }}`, email type HTML, message `{{ $json.html }}`.

 TIP

If no sales happened, the Filter outputs nothing and the email isn't sent. Prefer a "No sales today" email? Turn on **Always Output Data** on the Filter and add an If node checking for an empty result.

23.3 Recipe 3: nightly file backup

Goal: every night, copy new files from one cloud folder to another (for example Google Drive to Dropbox), and log what was copied.

Nodes: Schedule Trigger, Google Drive (search/list files), Data Table, Google Drive (download), Dropbox (upload), Data Table (insert).

BUILD THE BACKUP

- 1 Schedule Trigger:**
Days, at 2am.
- 2 Google Drive:**
list or search the files in your chosen folder.
- 3 Data Table > If Row Does Not Exist:**
check each file ID against a `backed_up` table (Chapter 15), so only new files continue.
- 4 Google Drive > Download:**
download each file by ID. It arrives as binary data.
- 5 Dropbox > Upload:**
set the path to `/Backups/{{ $json.name }}` and the binary property to the download's slot (usually `data`).

- 6 Data Table > Insert:**
record the file ID, name and `{{ $now.toISO() }}`.

Self-hosted variation: use **Read/Write Files from Disk** to save files to a folder on your server that your system backups already cover. Large files use memory, so add Loop Over Items with a small batch size if you back up many big files.

23.4 Recipe 4: social post scheduler

Goal: write posts in a spreadsheet with a date and time; n8n publishes each one when it's due.

Sheet columns: `post_text`, `image_url`, `publish_at` (e.g. `2026-10-20 09:00`), `status`.

Nodes: Schedule Trigger, Google Sheets, Filter, social node (LinkedIn, X, Facebook...), Google Sheets update.

BUILD THE SCHEDULER

- 1 Schedule Trigger:**
every 15 minutes.
- 2 Google Sheets:**
get rows from the “Posts” sheet.
- 3 Filter:**
`status` **is empty** AND `{{ $json.publish_at.toDateTime() }}` **is before** `{{ $now }}` (Date & Time comparison).
- 4 Social node:**
for example **LinkedIn > Create a post** with `{{ $json.post_text }}`. If there's an image, download it first with HTTP Request and attach the binary.
- 5 Google Sheets > Update row:**
set `status` to `posted` and add the time, matching on the row number.

WARNING

Always mark rows as posted *after* posting succeeds, and attach your error workflow. Otherwise a failure could cause a post to go out twice, or not at all.

Extend it: add a Basic LLM Chain that turns one long post into short versions for each network; add an approval step with a Wait node set to **On Form Submitted**.

- ☐ I've built at least one recipe end to end
- ☐ It uses Remove Duplicates or a data table so it never repeats work

Recipes: Business Workflows

These recipes handle the admin that eats into a small business's week: capturing leads, enriching them, processing invoices, keeping tools in sync and watching the web for changes.

IN THIS CHAPTER

- Recipe 5: form to CRM
- Recipe 6: lead enrichment
- Recipe 7: invoice processing with AI
- Recipe 8: Notion sync
- Recipe 9: web page change monitor

24.1 Recipe 5: form to CRM

Goal: a contact form that adds the person to your CRM, emails them a welcome, and pings you.

Nodes: n8n Form Trigger, Edit Fields, CRM node (HubSpot, Pipedrive, Airtable or a Google Sheet), Gmail, Slack or Telegram.

BUILD IT

- 1 n8n Form Trigger:**
title “Get in touch”; fields Name (Text, required), Email (Email, required), Company (Text), Message (Textarea). **Respond When:** Form Is Submitted.
- 2 Edit Fields:**
tidy the data: `name = {{ $json.Name.trim().toUpperCase() }}, email = {{ $json.Email.trim().toLowerCase() }}`.

- 3 **CRM node:**
choose the create-or-update contact operation (for example HubSpot **Contact > Create or Update**), mapping email, name and company. “Create or update” avoids duplicates when someone submits twice.
- 4 **Gmail:**
send a short welcome to `{{ $json.email }}`.
- 5 **Slack:**
post `New Lead: {{ $json.name }} ({{ $json.company }})` to `#sales`.

TIP

Spam filler? Add a **Hidden Field** to the form and drop any submission where it's filled in (bots fill everything), or run the message through a Text Classifier and discard `spam`.

24.2 Recipe 6: lead enrichment

Goal: when a new lead arrives, look up their company's website, summarise what it does, and score the lead.

Nodes: a trigger (Recipe 5's form, a CRM trigger, or a Sheets trigger), HTTP Request, HTML, Information Extractor, CRM update.

BUILD IT

- 1 **Get the domain:**
`{{ $json.email.extractDomain() }}`. Skip free mail domains with a Filter (`gmail.com`, `outlook.com` ...).
- 2 **HTTP Request:**
GET `https://{{ $json.domain }}`, response format **Text**. Turn on **Never Error** and Retry On Fail: some sites are slow.
- 3 **HTML > Extract HTML Content:**
pull the `title`, the meta description and the `body` text.
- 4 **Information Extractor:**
attributes `industry`, `company_size_guess`, `one_line_summary`, `fit_score` (1–5, with your ideal-customer description in the system prompt).
- 5 **CRM update:**
write the fields back to the contact; if `fit_score` ≥ 4 , also alert sales.

Extend it: use a dedicated enrichment API through HTTP Request; route low-fit leads to a nurture email sequence.

24.3 Recipe 7: invoice processing with AI

Goal: invoices emailed to `invoices@` are read automatically, logged in a sheet, and filed in Drive.

Nodes: Gmail Trigger (or IMAP Email trigger), Filter, Extract From File, Information Extractor, Google Drive, Google Sheets, If.

BUILD IT

- 1 Gmail Trigger:**
poll every few minutes for messages to your invoices address or with an “Invoices” label, downloading attachments.
- 2 Filter:**
keep items whose attachment is a PDF (`mimeType` is `application/pdf`).
- 3 Extract From File > Extract From PDF:**
point it at the attachment’s binary property. Out comes the invoice’s text.
- 4 Information Extractor:**
from the text, extract `supplier` , `invoice_number` , `invoice_date` , `due_date` , `currency` , `total` (number).
- 5 Google Drive > Upload:**
save the PDF into “Invoices/2026” named `{{ $json.supplier }} {{ $json.invoice_number }}.pdf` .
- 6 Google Sheets > Append Row:**
log the fields with the Drive link.
- 7 If:**
total above your approval limit → Slack your manager with the details.

⚠ DANGER

AI extraction is very good but not perfect. Keep a person in the loop for payments: have the workflow *log and alert*, not pay. Add a “checked” column you tick by hand.

💡 TIP

Scanned paper invoices have no text layer, so Extract From PDF returns nothing. Send the file to a vision-capable model or an OCR service instead.

24.4 Recipe 8: Notion sync

Goal: keep a Notion database of tasks in step with a Google Sheet (or another tool), so teammates can work wherever they like.

Nodes: Schedule Trigger, Notion (get database pages), Google Sheets (get rows), Compare Datasets, Notion create/update, Sheets update.

BUILD A ONE-WAY SYNC (SHEET → NOTION)

- 1 Connect Notion.**
Create an internal integration in Notion's developer settings, copy its secret into an n8n Notion credential, and **share your database with the integration** from the database's connections menu.
- 2 Schedule Trigger:**
every 30 minutes.
- 3 Google Sheets:**
get all rows (each needs a stable `task_id`).
- 4 Notion > Database Page > Get Many:**
get all pages from the database, with their `task_id` property.
- 5 Compare Datasets:**
compare on `task_id`. Its outputs separate items only in the sheet (new), in both but different (changed), and only in Notion.
- 6 Notion > Create:**
for new items. **Notion > Update:** for changed items, using the page ID from the Notion side.

NOTE

Two-way sync is much harder: you need to know which side changed most recently, and avoid ping-ponging. Start one-way; add the other direction only if you truly need it, using a “last edited” timestamp on both sides.

24.5 Recipe 9: web page change monitor

Goal: check a page (a price, a job board, a “sold out” notice) every hour and alert you when it changes.

Nodes: Schedule Trigger, HTTP Request, HTML, Data Table, If, Telegram or email.

BUILD IT

- 1 Schedule Trigger:**
every hour.
- 2 HTTP Request:**
GET the page, response format **Text**.
- 3 HTML > Extract HTML Content:**
extract just the part you care about with a **CSS selector** (right-click the element in your browser, **Inspect**, then copy the selector), into a field `value`.
- 4 Data Table > Get:**
read the last saved value for this URL.
- 5 If:**
new `value` is not equal to the saved one.
- 6 True branch:**
send `Changed: {{ $json.value }}` by Telegram, then **Data Table > Upsert** the new value.

⚠ WARNING

Respect websites: check politely (hourly, not every minute), follow their terms, and prefer an official API or RSS feed if one exists. Pages that build their content with JavaScript may need a scraping service rather than a plain HTTP request.

- ☐ Lead data is cleaned before it reaches my CRM
- ☐ AI-extracted invoice data is checked by a person before payment
- ☐ My Notion integration has access to the database
- ☐ My monitor checks politely and stores the last value

Recipes: AI Workflows

The last three recipes put AI to work: sorting your inbox, answering customers from your own documents, and a personal assistant you can message from your phone.

IN THIS CHAPTER

- Recipe 10: email triage with AI
- Recipe 11: AI support chatbot with RAG
- Recipe 12: Telegram assistant bot

25.1 Recipe 10: email triage with AI

Goal: every new email is classified, labelled, and, for customer questions, given a draft reply waiting for you to check.

Nodes: Gmail Trigger, Text Classifier, Gmail (add label), Basic LLM Chain with Structured Output Parser, Gmail (create draft), Slack.

BUILD IT

- 1 Gmail Trigger:**
event **Message Received**, polling every minute or five. Keep **Simplify** on.
- 2 Text Classifier:**
input `Subject: {{ $json.Subject }} Body: {{ $json.snippet }}`. Categories with short descriptions: `customer_question`, `invoice`, `newsletter`, `personal`, `spam`. Each category becomes an output.
- 3 Gmail > Add Label:**
on each output, add the matching label to `{{ $json.id }}`.

- 4 **On customer_question:**
a **Basic LLM Chain** with your tone and policies in the prompt, and a Structured Output Parser returning `urgency` (low/medium/high) and `reply`.
- 5 **Gmail > Create draft:**
a reply to the sender with the AI's `reply`. You review and send it yourself.
- 6 **If urgency is high:**
Slack yourself a link to the email.

TIP

Start with drafts, never automatic sending. After a few weeks of reading the drafts, you'll know which categories are safe to automate further.

NOTE

Every new email is one execution. A busy inbox can use thousands a month. Narrow the trigger with Gmail search filters (for example only the inbox, not promotions) to save executions and AI costs.

25.2 Recipe 11: AI support chatbot with RAG

Goal: a chat window, on its own page or your website, that answers customer questions from your help articles and hands over to a human when stuck.

Two workflows: a loader (documents → vector store) and the chatbot.

WORKFLOW A: LOAD YOUR HELP ARTICLES

- 1 **Trigger:**
Schedule (daily) or a Google Drive trigger on your “Help articles” folder.
- 2 **Google Drive:**
download the documents (or HTTP Request to fetch your help-centre pages).
- 3 **Vector store > Insert Documents:**
PGVector, Supabase, Qdrant or Pinecone, with an embeddings model, Default Data Loader and Recursive Character Text Splitter (Chapter 21). ::::checklist
 - Name every node clearly (“Get today’s orders,” not “HTTP Request3”)
 - Add **sticky notes** on the canvas explaining tricky parts
 - Test with pinned data before publishing
 - Set an error workflow
 - Watch your execution count on Cloud; use filters early to skip unnecessary runs

- Use **sub-workflows** (Execute Sub-workflow) for steps you repeat in many workflows :::

:::steps Workflow B: the chatbot

1. **Chat Trigger: Make Chat Publicly Available** on, mode **Hosted Chat** (or **Embedded Chat** for your site). Set a title and welcome message in the options.
2. **Guardrails (optional but wise): Check Text for Violations** with jailbreak and topical alignment checks; the Fail branch replies politely that you can only help with your products.
3. **AI Agent:** chat model, **Postgres Chat Memory** (or Simple Memory to start), and the vector store as a tool (**Retrieve Documents (As Tool for AI Agent)**) named `help_articles`.
4. **Add a hand-over tool:** a **Call n8n Workflow Tool** that runs a sub-workflow emailing your support inbox with the conversation and the customer's address. Describe it: "Use when you can't answer from `help_articles`, or the customer asks for a person."
5. **System message:** your shop's name, "answer only from `help_articles`," tone, length, and "never promise refunds; hand over instead."
6. **Test hard.** Ask easy questions, off-topic questions, and tricky ones. Read **Return Intermediate Steps** to see which tools it used.
7. **Publish** and share the chat URL, or embed it on your site. :::

☆ IMPORTANT

Tell customers they're talking to an AI, and give them an easy path to a human. Check your local rules on data protection before storing chat histories.

25.3 Recipe 12: Telegram assistant bot

Goal: message a private Telegram bot from your phone: "what's on tomorrow?", "add milk to my list", "summarise this link."

Nodes: Telegram Trigger, If (security check), AI Agent with memory and tools, Telegram (send message).

:::steps Build it

1. **Create the bot.** In Telegram, message **@BotFather**, send `/newbot`, follow the prompts, and copy the **bot token** into an n8n Telegram credential.
2. **Telegram Trigger:** updates **Message**. Send your bot a message while listening to see the data, and note your own `chat.id`.
3. **If:** continue only when `{{ $json.message.chat.id }}` equals your ID, so strangers can't use your assistant.
4. **AI Agent:** prompt `{{ $json.message.text }}`; chat model; Simple Memory with session key `{{ $json.message.chat.id }}`; tools such as Google Calendar (get events), Google Tasks or a data table (your list), and HTTP Request (fetch a link).

5. **Telegram > Send Message:** chat ID `{{ $('Telegram Trigger').item.json.message.chat.id }}`,
text `{{ $json.output }}`.

6. **Publish.** Telegram needs a public HTTPS address, so use Cloud or your server from Chapter 5. :::

 WARNING

Telegram allows only one webhook per bot. Testing in the editor while the workflow is published redirects messages to the test URL. Publish again after testing, or use a second bot for development.

Extend it: send voice notes and transcribe them with a speech-to-text model; add a daily morning message from a Schedule Trigger.

:::checklist

- My email triage creates drafts, not automatic replies
- My chatbot answers from my documents and can hand over to a human
- My Telegram bot only answers my own chat ID
- Each AI workflow has guardrails or approvals where it matters :::

Running Your Own n8n: Operations

If you self-host, you are now the person who keeps n8n healthy. The good news: with Docker Compose, the routine is short. This chapter covers backups, updates, configuration, scaling with queue mode, and security hardening.

:::summary In this chapter

- What to back up, and how to restore
- Updating safely
- Environment variables you'll actually use
- Keeping the database small
- Queue mode with Redis and workers
- A security hardening checklist
- Useful command-line tools :::

If you self-host for real (not just to experiment), cover these basics:

TOPIC	WHAT TO DO
Always-on hosting	Use a small cloud server (DigitalOcean, Hetzner, AWS...) or a service with one-click n8n, rather than your laptop
Docker Compose	n8n's recommended production setup, with a proper database (PostgreSQL)
HTTPS and a domain	Put n8n behind a reverse proxy with a certificate, and set <code>N8N_WEBHOOK_URL</code> to your public address so webhooks work
Encryption key	n8n encrypts credentials with a key in its data folder. Back it up; without it, saved credentials can't be decrypted
Backups	Back up the data volume and database regularly; export important workflows too
Updates	Update regularly for features and security fixes; read the release notes for breaking changes first
Access	Use a strong owner password and two-factor authentication; don't expose the editor to the internet unprotected

26.1 Backups

A complete backup has two parts:

- The `.n8n` folder** (in Docker, the `n8n_data` volume at `/home/node/.n8n`). It holds the `config` file with the encryption key and, with SQLite, the whole database.
- Your PostgreSQL database**, if you use one, backed up with Postgres's own tools.

Also keep a copy of your `.env` file and Compose file. Here's a simple nightly backup for the Chapter 5 setup, run from `~/n8n-compose`:

BACKUP.SH

```
#!/bin/sh
cd ~/n8n-compose
DAY=$(date +%F)
mkdir -p backups

# 1. Database dump
docker compose exec -T postgres \
  pg_dump -U n8n_admin n8n > backups/db-$DAY.sql

# 2. The .n8n folder (encryption key, settings)
docker run --rm -v n8n-compose_n8n_data:/data -v "$PWD/backups":/b \
  alpine tar czf /b/n8n-data-$DAY.tgz -C /data .

# 3. Keep 14 days
find backups -type f -mtime +14 -delete
```

Make it executable with `chmod +x backup.sh`, and schedule it with `crontab -e`, adding a line like `30 3 * * * /root/n8n-compose/backup.sh`. Docker names volumes after the project folder, so check yours with `docker volume ls`.

⚠ DANGER

A backup on the same server dies with the server. Copy the `backups` folder elsewhere regularly (another provider's storage, or your own computer), and test a restore at least once.

Exporting workflows and credentials

n8n's command line can export everything as readable JSON. In Docker, run commands with

```
docker compose exec -u node n8n:
```

TERMINAL

```
n8n export:workflow --backup --output=backups/workflows/  
n8n export:credentials --backup --output=backups/credentials/
```

To restore into a fresh instance, complete the owner setup first, then:

TERMINAL

```
n8n import:workflow --separate --input=backups/workflows/  
n8n import:credentials --separate --input=backups/credentials/
```

Imported workflows start unpublished. Credentials only decrypt if the new instance uses the **same encryption key**. CLI exports don't include users, execution history, variables or instance settings, so keep the full backup too.

26.2 Updating

n8n releases a new minor version most weeks. Update at least monthly so you never have to jump many versions at once.

steps Update the Chapter 5 setup

1. **Read the release notes** for anything marked as a breaking change.
2. **Back up** (run `backup.sh`).
3. **Change the version**. Edit `N8N_VERSION` in `.env` to the new stable version.
4. **Pull and restart:**

TERMINAL

```
docker compose pull  
docker compose up -d
```

5. Check it. `docker compose logs --tail=50 n8n`, then open the editor and look at a few recent executions. :::

To roll back, set the old version again and restart. If the update migrated the database, restore your database backup too.

26.3 Environment variables you'll actually use

:::compact

VARIABLE	DEFAULT	WHAT IT DOES
<code>N8N_WEBHOOK_URL</code>	–	Public base URL for webhooks (replaces <code>WEBHOOK_URL</code>)
<code>N8N_PROXY_HOPS</code>	0	Number of reverse proxies in front of n8n
<code>N8N_HOST</code> / <code>N8N_PROTOCOL</code>	localhost / http	Public host name and protocol
<code>N8N_EDITOR_BASE_URL</code>	–	Public URL of the editor, used in emails
<code>N8N_ENCRYPTION_KEY</code>	Random	Key that encrypts credentials
<code>GENERIC_TIMEZONE</code>	New York	Default time zone for schedules
<code>EXECUTIONS_DATA_PRUNE</code>	true	Delete old execution data automatically
<code>EXECUTIONS_DATA_MAX_AGE</code>	336	Hours to keep executions (14 days)
<code>EXECUTIONS_DATA_PRUNE_MAX_COUNT</code>	10000	Maximum executions kept
<code>EXECUTIONS_TIMEOUT</code>	-1	Default workflow timeout in seconds (off)
<code>N8N_CONCURRENCY_PRODUCTION_LIMIT</code>	-1	Cap on simultaneous production runs
<code>N8N_PAYLOAD_SIZE_MAX</code>	16	Max request size in MiB
<code>N8N_DATA_TABLES_MAX_SIZE_BYTES</code>	200 MiB	Data table storage limit
<code>N8N_BLOCK_ENV_ACCESS_IN_NODE</code>	false	Stop workflows reading environment variables
<code>N8N_RESTRICT_FILE_ACCESS_TO</code>	–	Folders file nodes may use
<code>NODES_EXCLUDE</code>	–	Nodes to disable entirely
<code>N8N_PUBLIC_API_DISABLED</code>	false	Turn off n8n's REST API
<code>N8N_DIAGNOSTICS_ENABLED</code>	true	Anonymous telemetry to n8n
:::		

Add variables to the `environment:` section of the n8n service (or to `.env` and reference them), then run `docker compose up -d`.

26.4 Keeping the database small

Every execution stores its data. Busy instances fill disks quickly. By default n8n prunes executions older than 14 days or beyond 10,000. To save more space, set workflows to not save successful production executions (workflow settings), keep failures for debugging, and lower

`EXECUTIONS_DATA_MAX_AGE`.

26.5 Scaling with queue mode

One n8n process handles a lot, but very busy instances can use **queue mode**: the main instance receives triggers and webhooks and puts jobs in a queue; separate **worker** processes run them.

PIECE	ROLE
Main	Editor, triggers, webhooks; creates jobs
Redis	The queue (message broker)
Workers	Run executions; add more to scale
PostgreSQL	Required; SQLite isn't supported in queue mode
Webhook processors (optional)	Extra processes that receive webhook traffic

The essentials:

- Set `EXECUTIONS_MODE=queue` on main and every worker, and point them at Redis with `QUEUE_BULL_REDIS_HOST` and `QUEUE_BULL_REDIS_PORT` (plus `QUEUE_BULL_REDIS_PASSWORD` if set).
- Give **every** worker the same `N8N_ENCRYPTION_KEY` and database settings as main.
- Start a worker with the `worker` command, for example a second service using the n8n image with `command: worker`, or `n8n worker --concurrency=5`. Worker concurrency defaults to 10; n8n recommends 5 or more.
- Simple Memory and binary data in filesystem mode don't work in queue mode; use database chat memory and S3 storage instead.

NOTE

Queue mode itself is free to use. Running several *main* instances (multi-main) and the workers view in settings need paid licences.

26.6 Security hardening checklist

:::checklist

- Owner and all users have strong passwords and 2FA (`N8N_MFA_ENFORCED_ENABLED=true` can enforce it)
- Only ports 22, 80 and 443 are open; n8n's port 5678 isn't exposed
- HTTPS everywhere, through Caddy, Traefik or a load balancer
- n8n and the server's operating system are updated monthly
- Backups run nightly, are copied off the server, and a restore has been tested
- Public webhooks, forms and chats use authentication where possible
- `N8N_BLOCK_ENV_ACCESS_IN_NODE=true` if others build workflows on your instance
- Risky nodes (Execute Command, SSH, file access) are excluded with `NODES_EXCLUDE` if unneeded
- Community nodes only from authors you trust
- `n8n audit` run occasionally to spot risky settings :::

26.7 Command-line toolbox

Run these inside the container with `docker compose exec -u node n8n <command>`:

COMMAND	DOES
<code>n8n audit</code>	Generate a security audit report
<code>n8n export:workflow --backup --output=...</code>	Export all workflows
<code>n8n publish:workflow --id=<ID></code>	Publish a workflow
<code>n8n unpublish:workflow --all</code>	Unpublish everything (emergency stop)
<code>n8n execute --id <ID></code>	Run a workflow once
<code>n8n user-management:reset</code>	Reset users, back to owner setup
<code>n8n mfa:disable --email=you@example.com</code>	Turn off 2FA for a locked-out user

What n8n Really Costs

n8n's pricing is simple once you know what's counted. This chapter sets out the plans, the hidden extras (AI and servers), and practical ways to keep your bill low.

summary In this chapter

- What counts as an execution
- Cloud plans, limits and prices
- Self-hosted costs, including the Business plan
- AI costs: Gateway credits and your own keys
- Ten ways to use fewer executions

27.1 What counts

An **execution** is one full run of a workflow, however many steps it has. Plans count **production executions**: runs started by triggers on published workflows. Manual test runs don't count, and sub-workflow executions don't count towards the monthly limit. For chat agents, each turn of the conversation is one execution.

SCHEDULE	EXECUTIONS PER MONTH (APPROX.)
Once a day	30
Every hour	730
Every 15 minutes	2,900
Every 5 minutes	8,700
Every minute	43,800

27.2 n8n Cloud plans

PLAN	YEARLY BILLING	MONTHLY BILLING	EXECUTIONS
Starter	€20/month	€24/month	2,500
Pro	€50/month	€60/month	10,000
Enterprise	Contact sales	Contact sales	Custom

Pro is also sold with more executions (for example 50,000 a month, with 50 concurrent runs). Prices from n8n.io/pricing, October 2026; yearly billing saves about 17%.

:::compact

LIMIT	STARTER	PRO	ENTERPRISE
Concurrent executions	5	20 or 50	200+
Max execution duration	5 min	40 min	40 min
Saved executions	2,500	25,000	50,000
Execution log retention	7 days	30 days	Unlimited
Shared projects	1	3	Unlimited
Assistant credits a month	1,600	Up to 9,600	On request
Workflow history	1 day	5 days	365+ days
:::			

All plans include unlimited users, unlimited workflows, every integration, AI agents and MCP. Time spent waiting (in a Wait node or for an approval) doesn't count towards the duration limit. Runs over your concurrency limit wait in a queue rather than failing.

27.3 Self-hosted costs

- **Community Edition:** free, unlimited executions. You pay only for the server: a small VPS typically costs a few euros a month, plus a domain name.
- **Business:** €667/month billed yearly (€800 monthly) for 40,000 executions, self-hosted only, adding SSO, environments, Git version control and more shared projects. n8n's Start-up Plan offers 50% off Business for companies with fewer than 20 employees, if you qualify.
- **Enterprise:** contact sales.

Don't forget your time: an hour a month for updates and checks is realistic.

27.4 AI costs

AI model use is billed separately from executions:

- **Gateway credits** (Cloud Starter and Pro): prepaid credit spent per request at the rates on n8n's service pricing page. When the balance hits zero, Gateway-powered nodes stop until you top up; top-ups expire after 12 months.
- **Your own API keys:** the provider bills you directly per token. Set a monthly spending limit in the provider's console.
- **Local models with Ollama** (self-hosted): no per-token cost, but you need capable hardware.

27.5 Ten ways to use fewer executions

:::checklist

- Use the slowest schedule that does the job
- Prefer instant (webhook) triggers to frequent polling where possible
- Filter early, in the trigger's own options if it has them
- Batch work: one run that handles 100 rows beats 100 runs
- Move shared steps into sub-workflows (they don't count)
- Stop chatty triggers, like every Gmail message, with narrower searches
- Unpublish workflows you no longer need
- Use Remove Duplicates or data tables to skip already-handled items
- Pick cheaper AI models for simple jobs like classification
- Check the Insights dashboard monthly to spot heavy workflows :::

NOTE

Prices and limits change. Check n8n.io/pricing before you buy; the figures here were correct in October 2026.

Templates and Community

You rarely need to start from scratch. Thousands of people have already built and shared workflows like yours, and a friendly community is ready to help when you're stuck.

summary In this chapter

- Using templates safely
- Community nodes
- Getting good help on the forum
- Where to keep learning

28.1 Templates

You rarely need to start from scratch. **n8n.io/workflows** hosts thousands of free templates: lead capture, social media posting, invoice processing, AI support agents, and more. Click **Use workflow** to copy one into your n8n, then add your own credentials.

steps Adopt a template safely

1. **Read the description and sticky notes.** Good templates explain what to change.
2. **Import it**, then open every node. Replace placeholder IDs, sheet names and channels with yours.
3. **Connect your credentials.** Templates never include anyone's secrets.
4. **Test with pinned or sample data** before publishing, especially any node that sends, posts or deletes.
5. **Rename it and add your error workflow.**

28.2 Community nodes

Self-hosted users can also install **community nodes**, extra nodes built by other users, from **Settings > Community nodes**. Install only well-maintained ones from authors you trust. A community node runs with the same access as n8n itself, so treat installing one like installing software on your server. On Cloud, a selection of verified community nodes is available from the nodes panel.

28.3 Getting help

The **community forum** (community.n8n.io) is friendly and active. Search it first; someone has usually solved your problem. When you post, include:

- your n8n version and whether you're on Cloud or self-hosted;
- what you expected and what happened instead;
- the exact error message;
- a small copy of the workflow (select the nodes, copy, and paste them in a code block), with personal data removed;
- a sample of the input data.

PROMPT

Help me write a clear n8n forum question. My setup: [Cloud / self-hosted, version]. Goal: [what the workflow should do]. Problem: [what happens]. Error: [paste]. Make it short, polite and easy to answer.

28.4 Keep learning

- **docs.n8n.io**: the official documentation, including step-by-step learning paths and a glossary.
- **Templates** for the apps you use: study how others built them.
- **The release notes**: a quick weekly skim shows new features.
- **This manual's recipes**: rebuild one with a twist each week.

:::checklist

- I've imported and adapted one template
- I check any community node's author and maintenance before installing
- I know what to include in a forum question :::

Troubleshooting

When something isn't working, find the closest symptom below. Most problems fall into a handful of patterns, and each has a quick fix.

29.1 Triggers and publishing

SYMPTOM	LIKELY CAUSE	FIX
Schedule never runs	Workflow not published	Publish it; check the button state
Runs at the wrong hour	Time zone	Set it in workflow settings or <code>GENERIC_TIMEZONE</code>
Webhook works in testing, not live	Unpublished, or app uses the test URL	Publish; give the app the production URL
Webhook returns 404	Wrong path, method or URL type	Check method (GET/POST) and <code>/webhook/</code> vs <code>/webhook-test/</code>
Edits have no effect live	Changes still in draft	Publish again
"Published, partial"	A trigger couldn't activate	Hover to see which, fix its credential, republish
Telegram stops after testing	One webhook per bot	Publish again after testing
Polling trigger misses items	Poll interval or filters	Check Poll Times and the trigger's filters
Form shows an error after submit	Workflow failed with Workflow Finishes	Check Executions for the failed run

29.2 Data and expressions

SYMPTOM	LIKELY CAUSE	FIX
Expression shows <code>undefined</code>	Wrong path or capitals	Drag the field in from the input panel
A node runs too many times	It received more items than expected	Check item counts; Aggregate or Limit first
Nodes after a step don't run	That step output zero items	Expected? If not, Always Output Data
"Referenced node is unexecuted"	Expression points at a node not on this path	Reference a node that ran, or use Merge
Paired item error	Item links broken	Use <code>.first()</code> / <code>.all()</code> , or fix the Code node
<code>[object Object]</code> in text	Inserted a whole object	Pick a field, or <code>JSON.stringify()</code>
Dates off by hours	Time zone or UTC strings	Set workflow time zone; use <code>.toDateTIme()</code>
Merge outputs nothing	One input received no items	Check both branches ran
File missing in next node	Wrong binary property name	Look at the output's binary slot name

29.3 Credentials and APIs

SYMPTOM	LIKELY CAUSE	FIX
401 Unauthorized	Key wrong or OAuth token expired	Reconnect or replace the credential
403 Forbidden	Missing permission or scope	Grant the scope; reconnect OAuth
429 Too Many Requests	Rate limit	Batching, Loop Over Items + Wait, Retry On Fail
Google "access denied"	App in Testing mode	Add yourself as a test user
Google redirect mismatch	Wrong redirect URI	Paste n8n's exact OAuth Redirect URL
OAuth redirects to <code>localhost</code>	Public URL not set (self-hosted)	Set <code>N8N_EDITOR_BASE_URL</code> , <code>N8N_WEBHOOK_URL</code>
Notion "object not found"	Database not shared with integration	Add the integration to the database
"Credentials could not be decrypted"	Encryption key changed	Restore the original <code>N8N_ENCRYPTION_KEY</code>

29.4 AI

SYMPTOM	LIKELY CAUSE	FIX
Agent won't use a tool	Vague tool name or description	Describe exactly when to use it
Agent invents facts	No grounding	Use RAG; "say you don't know"
Forgets earlier messages	No memory or session key changes	Attach memory; check the session key
Agent loops or stops early	Max Iterations	Clarify the task; adjust the limit
Output not in the format needed	Free text	Require Specific Output Format + parser
Knowledge base empty after restart	Simple Vector Store	Use a persistent vector store
Gateway nodes stop working	Credit balance is zero	Top up, or switch to your own key

29.5 Self-hosting

SYMPTOM	LIKELY CAUSE	FIX
Site unreachable	DNS, firewall, or container down	<code>ping</code> , <code>ufw status</code> , <code>docker compose ps</code>
Certificate error	Port 80 closed or DNS wrong	Open 80; check <code>docker compose logs caddy</code>
n8n restarting in a loop	Database settings	<code>docker compose logs n8n</code> ; check <code>.env</code>
Webhook URLs show <code>localhost</code>	<code>N8N_WEBHOOK_URL</code> not set	Set it and restart
Disk full	Execution data	Lower retention; prune; stop saving successes
Out of memory	Big files or huge item lists	Batches, sub-workflows, a bigger server
Lost data after update	Volume removed with <code>down -v</code>	Restore from backup; never use <code>-v</code> casually
Locked out by 2FA	Lost phone	Recovery codes, or <code>n8n mfa:disable --email=...</code>

TIP

Still stuck? Open the failed execution, find the *first* node whose output looks wrong, and ask on the forum with that node's input and error. Chapter 18 has the full routine.

Five Complete Projects

Each project is a full build, from an empty canvas to a published workflow. Work through them in order, because each one adds a new skill.

summary In this chapter

- Project 1: RSS news digest by email (schedules, loops, aggregation)
- Project 2: Web form to a spreadsheet (form trigger, mapping)
- Project 3: AI email summariser (AI nodes, filters)
- Project 4: A tiny API with a webhook (webhooks, responses)
- Project 5: Error alerts for all your workflows (Error Trigger)

30.1 Project 1: Daily RSS digest

steps

1. **Add a Schedule Trigger.** Set it to every day at 07:00. Check the workflow's time zone in `Workflow settings` > `Timezone`.
2. **Add an RSS Read node.** Paste a feed URL, such as a news site's RSS feed. Click **Execute step**. You'll get one item per article.
3. **Add a Limit node.** Keep the first 10 items.
4. **Add an Aggregate node.** Choose *Individual fields*, field `title`, and also `link`. Now you have one item holding lists.
5. **Add a Code node** to build the email body:

JAVASCRIPT

```
const t = $json.title, l = $json.link;
const html = t.map((x, i) => `<li><a href="${l[i]}">${x}</a></li>`).join('');
return [{ json: { html: `<ul>${html}</ul>` } }];
```

6. **Add a Send Email (or Gmail) node.** Set the subject to `Morning digest {{ $now.toFormat('yyyy-MM-dd') }}`, choose HTML, and set the body to `{{ $json.html }}`.
7. **Test the whole workflow**, then **Publish** it. :::

30.2 Project 2: Form to spreadsheet

:::steps

1. **Add an n8n Form Trigger.** Add fields: Name (text, required), Email (email), Message (textarea).
2. **Open the test URL** and submit a test entry.
3. **Add a Google Sheets node** (Append row). Connect your Google credential, then choose the spreadsheet and sheet.
4. **Map columns** by dragging `Name`, `Email`, and `Message` from the input panel.
5. **Add a “thank you” step.** Use the form’s completion settings to show a message.
6. **Publish** and share the **production** form URL. :::

30.3 Project 3: AI email summariser

:::steps

1. **Trigger:** a Gmail Trigger (or Email Trigger (IMAP)) that polls for new mail.
2. **Filter node:** keep only messages where the sender matches the people you care about.
3. **AI step:** add a Basic LLM Chain (or an OpenAI / Anthropic node) with the prompt:

PROMPT

Summarise this email in 2 sentences, then list any action items with due dates. Email: {{ \$json.text }}

4. **Send the summary** to yourself with Telegram, Slack, or email.
5. **Publish.** Watch the first few executions in the **Executions** tab. :::

⚠ WARNING

Emails can contain prompt injection, meaning text written to manipulate the AI. In this workflow the AI only produces a summary for *you*. Don't give an email-reading AI agent tools that send, delete, or pay without a human approval step.

30.4 Project 4: A tiny API

:::steps

1. **Add a Webhook node.** Method `POST`, path `greet`, and set *Respond to* to **Using ‘Respond to Webhook’ node**.
2. **Add an Edit Fields (Set) node.** Create `message` = `Hello, {{ $json.body.name }}`!
3. **Add a Respond to Webhook node.** Respond with the first incoming item as JSON.
4. **Test it.** Click *Listen for test event*, then from a terminal:

TERMINAL

```
curl -X POST <test-url> -H "Content-Type: application/json" -d '{"name":"Dylan"}'
```

5. **Publish** and use the production URL. Add header authentication on the Webhook node before you share it. :::

30.5 Project 5: Error alerts

:::steps

1. **Create a new workflow** called “Error handler.”
2. **Add an Error Trigger node.**
3. **Add a notification node** with: `Workflow {{ $json.workflow.name }} failed: {{ $json.execution.error.message }}` and the link `{{ $json.execution.url }}`.
4. **Save it.** It doesn’t need publishing to act as an error workflow.
5. **In each important workflow**, open `Workflow settings` > `Error workflow` and choose “Error handler.” :::

Node and Expression Reference

A desk reference for the nodes and expressions you’ll use most. Node names change a little between versions. Search the nodes panel if one isn’t exactly where this book says.

summary In this chapter

- Core nodes by job
- Expression variables
- Common data-transformation snippets
- Cron cheat sheet

31.1 Core nodes

compact

node	job
Manual Trigger	Run by clicking, for testing
Schedule Trigger	Run on an interval or cron schedule
Webhook	Run when a URL receives a request
n8n Form Trigger	Hosted form that starts a workflow
Chat Trigger	Chat window that starts a workflow (often with an AI Agent)
Error Trigger	Runs when another workflow fails
Execute Sub-workflow / Execute Sub-workflow Trigger	Call one workflow from another

HTTP Request	Call any web API
Code	Run JavaScript (or Python) on items
Edit Fields (Set)	Add, rename, or set fields
If	Split into true and false branches
Switch	Route to many branches by rules
Filter	Drop items that don't match
Merge	Combine two inputs (append, by key, by position...)
Split Out	Turn a list inside one item into many items
Aggregate	Turn many items into one item holding lists
Summarize	Count, sum, or average, grouped by field
Loop Over Items	Process items in batches
Wait	Pause for a time or until a webhook call
Sort / Limit / Remove Duplicates	Tidy lists
Date & Time	Format and calculate dates
Respond to Webhook	Send a custom reply to a webhook caller
Send Email	Send mail through SMTP
AI Agent	An LLM that can use tools you attach
Basic LLM Chain	A single prompt → response step
...	

31.2 Expression variables

Expressions go inside `{{ }}`.

compact

EXPRESSION	MEANING
<code>\$json.field</code>	A field on the current item
<code>\$json["field with spaces"]</code>	A field with spaces in its name
<code>\$('Node Name').item.json.field</code>	The matching item from an earlier node
<code>\$('Node Name').first().json</code>	The first item from an earlier node
<code>\$input.all()</code>	All input items (Code node)
<code>\$input.first()</code>	The first input item

<code>\$now / \$today</code>	Current date-time / today at midnight (Luxon)
<code>\$now.toFormat('yyyy-MM-dd')</code>	Format a date
<code>\$now.minus({ days: 7 })</code>	Date arithmetic
<code>\$execution.id</code>	This run's ID
<code>\$workflow.name</code>	This workflow's name
<code>\$vars.NAME</code>	A workspace variable (on plans that have variables)
<code>\$env.NAME</code>	An environment variable (self-hosted, if allowed)
<code>{{ \$json.price * 1.2 }}</code>	Any JavaScript expression
<code>{{ \$json.name ?? 'unknown' }}</code>	Fallback when a value is missing
<code>:::</code>	

31.3 Code node snippets

ADD A FIELD TO EVERY ITEM

```
return $input.all().map(i => ({ json: { ...i.json, total: i.json.qty * i.json.price } }));
```

KEEP ONLY ITEMS WITH AN EMAIL

```
return $input.all().filter(i => i.json.email);
```

GROUP BY A FIELD

```
const groups = {};
for (const i of $input.all()) (groups[i.json.team] ??= []).push(i.json.name);
return Object.entries(groups).map(([team, names]) => ({ json: { team, names } }));
```

31.4 Cron cheat sheet

Fields are: minute, hour, day of month, month, day of week.

:::compact

EXPRESSION	RUNS
0 7 * * *	Every day at 07:00
*/15 * * * *	Every 15 minutes
0 9 * * 1-5	Weekdays at 09:00
0 0 1 * *	First day of every month at midnight
30 18 * * 0	Sundays at 18:30
:::	

Troubleshooting and FAQ

:::compact

PROBLEM	CAUSE	FIX
Workflow never runs on schedule	It isn't published	Publish it. Check the time zone setting
Webhook returns 404	Using the test URL while not listening, or the production URL while not published	Use the right URL for the state
"Referenced node is unexecuted"	An expression points to a node that didn't run in this branch	Run the earlier node, or guard with <code>?.</code>
Only the first item is processed	A node is set to "Execute once"	Turn off Execute once in the node settings
Credentials error	Expired OAuth or wrong scope	Reconnect the credential. Check its scopes
Rate-limit errors (429)	Calling an API too fast	Use Loop Over Items with a Wait. Enable <i>Retry on fail</i>
Code node "cannot read properties of undefined"	A field is missing on some items	Use <code>?.</code> and <code>??</code> defaults
Self-hosted data lost after restart	No persistent volume	Mount a volume for <code>/home/node/.n8n</code>
Webhooks unreachable when self-hosted	No public URL	Set <code>WEBHOOK_URL</code> and use a reverse proxy or tunnel
:::		

Do I need to know how to code? No. Code nodes help, and AI can write them for you.

Cloud or self-hosted? Cloud is easiest and handles updates and uptime. Self-hosting is cheaper at scale and keeps data on your server, but you handle backups, security, and updates.

How do I back up workflows? Download them as JSON (from the workflow menu), or use the n8n CLI `export:workflow` command on self-hosted instances. Keep the files in a private Git repository.

Can n8n run AI agents unattended? Yes. Give agents only the tools they need, add human approval for risky actions, and set error workflows. See Vol. 8 for autonomy settings.

Self-Hosting Environment Variable Reference

If you self-host n8n, you configure it with **environment variables**. This chapter lists the documented variables for the areas beginners touch most. It was generated from the official n8n docs (the *Environment variables* section of docs.n8n.io, source at github.com/n8n-io/n8n-docs) in October 2026. Defaults and names can change between versions, so check the docs for your version before relying on a value.

summary In this chapter

- How to set variables (Docker and Docker Compose)
- Variables by category, with type, default, and meaning

33.1 Setting variables

DOCKER-COMPOSE.YML (EXCERPT)

```
services:
  n8n:
    image: docker.n8n.io/n8nio/n8n
    environment:
      - GENERIC_TIMEZONE=America/Indiana/Indianapolis
      - TZ=America/Indiana/Indianapolis
      - WEBHOOK_URL=https://n8n.example.com/
    volumes:
      - n8n_data:/home/node/.n8n
volumes:
  n8n_data:
```

33.2 Deployment

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>HTTP_PROXY</code>	String	-	A URL to proxy unencrypted HTTP requests through.
<code>HTTPS_PROXY</code>	String	-	A URL to proxy TLS/SSL encrypted HTTP requests through.
<code>ALL_PROXY</code>	String	-	A URL to proxy both unencrypted and encrypted HTTP requests through.
<code>NO_PROXY</code>	String	-	A comma-separated list of hostnames or URLs that should bypass the proxy.
<code>N8N_ENFORCE_GLOBAL_USER_AGENT</code>	Boolean	<code>false</code>	When set to <code>true</code> , n8n replaces the default bare <code>n8n</code> User-Agent string with an RFC-compliant value ('Mozilla/5.0...
<code>N8N_GLOBAL_USER_AGENT_VALUE</code>	String	-	A custom User-Agent string to use for all outbound HTTP requests.
<code>N8N_EDITOR_BASE_URL</code>	String	-	Public URL where users can access the editor.
<code>N8N_DISABLE_UI</code>	Boolean	<code>false</code>	Set to <code>true</code> to disable the UI.
<code>N8N_PREVIEW_MODE</code>	Boolean	<code>false</code>	Set to <code>true</code> to run in preview mode.
<code>N8N_CANVAS_ONLY</code>	Boolean	<code>false</code>	Set to <code>true</code> to run n8n in canvas-only mode, hiding the header, sidebar, and overlay elements when viewing an individual...
<code>N8N_TEMPLATES_ENABLED</code>	Boolean	<code>true</code>	Enables workflow templates[^1] (true) or disable (false).
<code>N8N_TEMPLATES_HOST</code>	String	<code>https://api.n8n.io</code>	Change this if creating your own workflow template library.
<code>N8N_ENCRYPTION_KEY</code>	String	Random key generated by n8n	Provide a custom key used to encrypt credentials in the n8n database.
<code>N8N_ENV_FEAT_ENCRYPTION_KEY_ROTATION</code>	Boolean	<code>false</code>	Set to <code>true</code> on all instances (main and workers) to enable encryption key rotation.

<code>N8N_ENV_FEAT_OAUTH2_JWE</code>	Boolean	<code>false</code>	Set to <code>true</code> on all instances (main and workers) to enable JWE token decryption for OAuth 2.0 credentials.
<code>N8N_ENV_FEAT_TOKEN_EXCHANGE</code>	Boolean	<code>false</code>	Set to <code>true</code> to enable token exchange for embedding partners.
<code>N8N_OAUTH_JWE_JWKS_PER_MINUTE</code>	Number	<code>60</code>	Per-IP rate limit on the public JWKS endpoint (<code>/rest/.well-known/jwks.json</code>) used by JWE token decryption for OAuth 2.0...
<code>N8N_MCP_BASE_URL</code>	String	-	Public base URL where MCP clients reach the instance-level MCP server, when it differs from the instance base URL. Useful for...
<code>N8N_MCP_SERVER_RATE_LIMIT</code>	Number	<code>100</code>	Maximum number of requests per IP, per 5 minutes, to the MCP server endpoint (<code>/mcp-server/http</code>).
<code>N8N_OAUTH_SERVER_REGISTER_RATE_LIMIT</code>	Number	<code>10</code>	Maximum number of requests per IP, per 5 minutes, to the OAuth server client registration endpoint (<code>/oauth/register</code> and...
<code>N8N_OAUTH_SERVER_AUTHORIZE_RATE_LIMIT</code>	Number	<code>50</code>	Maximum number of requests per IP, per 5 minutes, to the OAuth server authorization endpoint (<code>/oauth/authorize</code> and...
<code>N8N_OAUTH_SERVER_TOKEN_RATE_LIMIT</code>	Number	<code>20</code>	Maximum number of requests per IP, per 5 minutes, to the OAuth server token endpoint (<code>/oauth/token</code> and <code>/mcp-oauth/token</code>).
<code>N8N_OAUTH_SERVER_REVOKE_RATE_LIMIT</code>	Number	<code>30</code>	Maximum number of requests per IP, per 5 minutes, to the OAuth server token revocation endpoint (<code>/oauth/revoke</code> and...
<code>N8N_USER_FOLDER</code>	String	<code>user-folder</code>	Provide the path where n8n will create the <code>.n8n</code> folder.
<code>N8N_PATH</code>	String	<code>/</code>	The path n8n deploys to.
<code>N8N_HOST</code>	String	<code>localhost</code>	Host name n8n runs on.
<code>N8N_PORT</code>	Number	<code>5678</code>	The HTTP port n8n runs on.
<code>N8N_LISTEN_ADDRESS</code>	String	<code>::</code>	The IP address n8n should listen on.
<code>N8N_PROTOCOL</code>	Enum string: <code>http</code> , <code>https</code>	<code>http</code>	The protocol used to reach n8n.

<code>N8N_SSL_KEY</code>	String	-	The SSL key for HTTPS protocol.
<code>N8N_SSL_CERT</code>	String	-	The SSL certificate for HTTPS protocol.
<code>N8N_PERSONALIZATION_ENABLED</code>	Boolean	<code>true</code>	Whether to ask users personalisation questions and then customise n8n accordingly.
<code>N8N_VERSION_NOTIFICATIONS_ENABLED</code>	Boolean	<code>true</code>	When enabled, n8n sends notifications of new versions and security updates.
<code>N8N_VERSION_NOTIFICATIONS_ENDPOINT</code>	String	<code>`https://api.n8n.io/versi...</code>	The endpoint to retrieve where version information.
<code>N8N_VERSION_NOTIFICATIONS_INFO_URL</code>	String	<code>`https://docs.n8n.io/gett...</code>	The URL displayed in the New Versions panel for more information.
<code>N8N_DIAGNOSTICS_ENABLED</code>	Boolean	<code>true</code>	Whether to share selected, anonymous telemetry with n8n.
<code>N8N_DIAGNOSTICS_CONFIG_FRONTEND</code>	String	<code>`1zPn9bgWPzlQc0p8Gj1uiK6D...</code>	Telemetry configuration for the frontend.
<code>N8N_DIAGNOSTICS_CONFIG_BACKEND</code>	String	<code>`1zPn7YoGC3ZXE9zLeTKLuQCB...</code>	Telemetry configuration for the backend.
<code>N8N_PUSH_BACKEND</code>	String	<code>websocket</code>	Choose whether the n8n backend uses server-sent events (<code>sse</code>) or WebSockets (<code>websocket</code>) to send changes to the UI.
<code>VUE_APP_URL_BASE_API</code>	String	<code>http://localhost:5678/</code>	Used when building the <code>n8n-editor-ui</code> package manually to set how the frontend can reach the backend API. Refer to Configure...
<code>N8N_HIRING_BANNER_ENABLED</code>	Boolean	<code>true</code>	Whether to show the n8n hiring banner in the console (true) or not (false).

<code>N8N_PUBLIC_API_SWAGGERUI_DISABLED</code>	Boolean	<code>false</code>	Whether the Swagger UI (API playground) is disabled (true) or not (false).
<code>N8N_PUBLIC_API_DISABLED</code>	Boolean	<code>false</code>	Whether to disable the public API (true) or not (false).
<code>N8N_PUBLIC_API_ENDPOINT</code>	String	<code>api</code>	Path for the public API endpoints.
<code>N8N_GRACEFUL_SHUTDOWN_TIMEOUT</code>	Number	<code>30</code>	How long should the n8n process wait (in seconds) for components to shut down before exiting the process.
<code>N8N_DEV_RELOAD</code>	Boolean	<code>false</code>	When working on the n8n source code, set this to <code>true</code> to automatically reload or restart the application when changes occur...
<code>N8N_REINSTALL_MISSING_PACKAGES</code>	Boolean	<code>false</code>	If set to <code>true</code> , n8n will automatically attempt to reinstall any missing packages.
<code>N8N_TUNNEL_SUBDOMAIN</code>	String	-	Specifies the subdomain for the n8n tunnel.
<code>N8N_PROXY_HOPS</code>	Number	<code>0</code>	Number of reverse-proxies n8n is running behind.
...			

33.3 Endpoints

...compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>N8N_PAYLOAD_SIZE_MAX</code>	Number	<code>16</code>	The maximum payload size in MiB. The default is 16 MiB, but this can be increased by adjusting the variable to a higher value.
<code>N8N_FORMDATA_FILE_SIZE_MAX</code>	Number	<code>200</code>	Max payload size for files in form-data webhook payloads in MiB.
<code>N8N_METRICS</code>	Boolean	<code>false</code>	Whether to enable the <code>/metrics</code> endpoint.
<code>N8N_METRICS_PREFIX</code>	String	<code>n8n_</code>	Optional prefix for n8n specific metrics names.
<code>N8N_METRICS_INCLUDE_DEFAULT_METRICS</code>	Boolean	<code>true</code>	Whether to expose default system and node.js metrics.
<code>N8N_METRICS_INCLUDE_CACHE_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics (true) for cache hits and misses, or not include them (false).
<code>N8N_METRICS_INCLUDE_MESSAGE_EVENT_BUS_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics (true) for events, or not include them (false).
<code>N8N_METRICS_INCLUDE_WORKFLOW_ID_LABEL</code>	Boolean	<code>false</code>	Whether to include a label for the workflow ID on workflow metrics.

<code>N8N_METRICS_INCLUDE_NODE_TYPE_LABEL</code>	Boolean	<code>false</code>	Whether to include a label for the node type on node metrics.
<code>N8N_METRICS_INCLUDE_CREDENTIAL_TYPE_LABEL</code>	Boolean	<code>false</code>	Whether to include a label for the credential type on credential metrics.
<code>N8N_METRICS_INCLUDE_API_ENDPOINTS</code>	Boolean	<code>false</code>	Whether to expose metrics for API endpoints.
<code>N8N_METRICS_INCLUDE_API_PATH_LABEL</code>	Boolean	<code>false</code>	Whether to include a label for the path of API invocations.
<code>N8N_METRICS_INCLUDE_API_METHOD_LABEL</code>	Boolean	<code>false</code>	Whether to include a label for the HTTP method (GET, POST, ...) of API invocations.
<code>N8N_METRICS_INCLUDE_API_STATUS_CODE_LABEL</code>	Boolean	<code>false</code>	Whether to include a label for the HTTP status code (200, 404, ...) of API invocations.
<code>N8N_METRICS_INCLUDE_QUEUE_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics for jobs in scaling mode.
<code>N8N_METRICS_QUEUE_METRICS_INTERVAL</code>	Integer	<code>20</code>	How often (in seconds) to update queue metrics.
<code>N8N_METRICS_INCLUDE_SCHEDULER_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics for the durable scheduler, such as queue depth, scheduling lag, dispatch throughput, retries, and...
<code>N8N_METRICS_SCHEDULER_INTERVAL</code>	Integer	<code>20</code>	How often (in seconds) to refresh the queue-health gauges for the durable scheduler.
<code>N8N_METRICS_INCLUDE_POLL_TRIGGER_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics for trigger nodes that poll, such as poll duration, poll errors, overlapping polls, and cursor commits.
<code>N8N_METRICS_INCLUDE_SYSTEM_TASK_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics for n8n's internal maintenance tasks, such as run duration, last success, skipped occurrences, and...
<code>N8N_METRICS_INCLUDE_SSRF_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics for SSRF protection checks.
<code>N8N_METRICS_INCLUDE_DNS_CACHE_METRICS</code>	Boolean	<code>false</code>	Whether to include metrics for the DNS cache (currently only used by SSRF protection).
<code>N8N_ENDPOINT_REST</code>	String	<code>rest</code>	The path used for REST endpoint.
<code>N8N_ENDPOINT_WEBHOOK</code>	String	<code>webhook</code>	The path used for webhook endpoint.

<code>N8N_ENDPOINT_WEBHOOK_TEST</code>	String	<code>webhook-test</code>	The path used for test-webhook endpoint.
<code>N8N_ENDPOINT_WEBHOOK_WAIT</code>	String	<code>webhook-waiting</code>	The path used for waiting-webhook endpoint.
<code>N8N_ENDPOINT_HEALTH</code>	String	<code>healthz</code>	The path used for health check endpoint.
<code>N8N_WEBHOOK_URL</code>	String	-	Sets the base URL for both test and production webhooks when running n8n behind a reverse proxy.
<code>WEBHOOK_URL</code>	String	-	Deprecated from n8n 2.35.0; alias of <code>N8N_WEBHOOK_URL</code> .
<code>N8N_DISABLE_PRODUCTION_MAIN_PROCESS</code>	Boolean	<code>false</code>	Disable production webhooks from main process.
...			

33.4 Executions

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>EXECUTIONS_MODE</code>	Enum string: <code>regular</code> , <code>queue</code>	<code>regular</code>	Whether executions should run directly or using queue. Refer to Queue mode for more details.
<code>EXECUTIONS_TIMEOUT</code>	Number	<code>-1</code>	Sets a default timeout (in seconds) to all workflows after which n8n stops their execution.
<code>EXECUTIONS_TIMEOUT_MAX</code>	Number	<code>3600</code>	The maximum execution time (in seconds) that users can set for an individual workflow.
<code>N8N_AI_TIMEOUT_MAX</code>	Number	<code>3600000</code>	Sets the HTTP request timeout in milliseconds for AI and LLM nodes (such as OpenAI, Anthropic, Mistral, and Ollama).
<code>EXECUTIONS_DATA_SAVE_ON_ERROR</code>	Enum string: <code>all</code> , <code>none</code>	<code>all</code>	Whether n8n saves execution data on error.
<code>EXECUTIONS_DATA_SAVE_ON_SUCCESS</code>	Enum string: <code>all</code> , <code>none</code>	<code>all</code>	Whether n8n saves execution data on success.
<code>EXECUTIONS_DATA_SAVE_ON_PROGRESS</code>	Boolean	<code>false</code>	Whether to save progress for each node executed (true) or not (false).
<code>EXECUTIONS_DATA_SAVE_MANUAL_EXECUTIONS</code>	Boolean	<code>true</code>	Whether to save data of executions when started manually.

<code>N8N_EXECUTION_DATA_STORAGE_MODE</code>	Enum string: <code>database</code> , <code>filesystem</code> , <code>s3</code> , <code>azure</code>	<code>database</code>	Where n8n stores execution data.
<code>N8N_STORAGE_PATH</code>	String	<code>N8N_USER_FOLDER/storage</code>	Base path for filesystem storage.
<code>EXECUTIONS_DATA_PRUNE</code>	Boolean	<code>true</code>	Whether to delete data of past executions on a rolling basis.
<code>EXECUTIONS_DATA_MAX_AGE</code>	Number	<code>336</code>	The execution age (in hours) before it's deleted.
<code>EXECUTIONS_DATA_PRUNE_MAX_COUNT</code>	Number	<code>10000</code>	Maximum number of executions to keep in the database. 0 = no limit.
<code>EXECUTIONS_DATA_HARD_DELETE_BUFFER</code>	Number	<code>1</code>	How old (hours) the finished execution data has to be to get hard-deleted.
<code>EXECUTIONS_DATA_PRUNE_HARD_DELETE_INTERVAL</code>	Number	<code>15</code>	How often (minutes) execution data should be hard-deleted.
<code>EXECUTIONS_DATA_PRUNE_SOFT_DELETE_INTERVAL</code>	Number	<code>60</code>	How often (minutes) execution data should be soft-deleted.

<code>EXECUTIONS_DATA_MAX_DISPLAY_SIZE</code>	Number	<code>104857600</code>	Maximum size (in bytes) of execution data that n8n loads when displaying an execution (editor, execution detail, and public API).
<code>N8N_CONCURRENCY_PRODUCTION_LIMIT</code>	Number	<code>-1</code>	Max production executions allowed to run concurrently, in both regular and scaling modes. <code>-1</code> to disable in regular mode.
<code>N8N_CONCURRENCY_EVALUATION_LIMIT</code>	Number	License-tier default	Max test cases that can run in parallel within a single evaluation test run.
<code>N8N_WORKFLOW_AUTODEACTIVATION_ENABLED</code>	Boolean	<code>false</code>	Whether workflows are automatically unpublished after repeated crashed executions.
<code>N8N_WORKFLOW_AUTODEACTIVATION_MAX_LAST_EXECUTIONS</code>	Number	<code>3</code>	Number of crashed executions before unpublishing a workflow.
...			

33.5 Security

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>N8N_BLOCK_ENV_ACCESS_IN_NODE</code>	Boolean	<code>false</code>	Whether to allow users to access environment variables in expressions and the Code node (false) or not (true).
<code>N8N_BLOCK_FILE_ACCESS_TO_N8N_FILES</code>	Boolean	<code>true</code>	Set to <code>true</code> to block access to all files in the <code>.n8n</code> directory and user defined configuration files.
<code>N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS</code>	Boolean	<code>false</code>	Set to <code>true</code> to try to set 0600 permissions for the settings file, giving only the owner read and write access.

<code>N8N_RESTRICT_FILE_ACCESS_TO</code>	String		Limits access to files in these directories.
<code>N8N_SECURITY_AUDIT_DAYS_ABANDONED_WORKFLOW</code>	Number	90	Number of days to consider a workflow abandoned if it's not executed.
<code>N8N_CONTENT_SECURITY_POLICY</code>	String	<code>`script-src`stri...</code>	The Content-Security-Policy n8n serves on its own pages.
<code>N8N_CONTENT_SECURITY_POLICY_REPORT_ONLY</code>	String		The policy n8n serves as <code>Content-Security-Policy-Report-Only</code> , in the same formats as <code>N8N_CONTENT_SECURITY_POLICY</code> .
<code>N8N_SECURE_COOKIE</code>	Boolean	<code>true</code>	Ensures that cookies are only sent over HTTPS, enhancing security.
<code>N8N_SAMESITE_COOKIE</code>	Enum string: <code>strict</code> , <code>lax</code> , <code>none</code>	<code>lax</code>	Controls cross-site cookie behavior (learn more): <code>strict</code> : Sent only for first-party requests. <code>lax</code> (default): Sent with...
<code>N8N_GIT_NODE_DISABLE_BARE_REPOS</code>	Boolean	<code>false</code>	Set to <code>true</code> to prevent the Git node from working with bare repositories, enhancing security.
<code>N8N_GIT_NODE_ENABLE_HOOKS</code>	Boolean	<code>false</code>	Set to <code>true</code> to allow the Git node to execute Git hooks.
<code>N8N_POSTMESSAGE_ALLOWED_ORIGINS</code>	String		Origins allowed to exchange <code>postMessage</code> commands with the editor when it's embedded in another page.
...			

33.6 Database

compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>DB_TYPE</code> <code>/_FILE</code>	Enum string: <code>sqlite</code> , <code>postgresdb</code>	<code>sqlite</code>	The database to use.
<code>DB_TABLE_PREFIX</code>	*	-	Prefix to use for table names.
<code>DB_PING_INTERVAL_SECONDS</code>	Number	2	How often, in seconds, n8n pings the database to check that the connection is still alive.
<code>DB_PING_TIMEOUT_MS</code>	Number	5000	How long, in milliseconds, n8n waits for a single ping to respond before counting it as a failure.

<code>DB_PING_MAX_FAILURES_BEFORE_RECOVERY</code>	Number	3	How many pings in a row must fail before n8n treats the connection as lost and starts recovery.
<code>DB_RECOVERY_BACKOFF_MIN_MS</code>	Number	1000	How long, in milliseconds, n8n waits before its first recovery attempt.
<code>DB_RECOVERY_BACKOFF_MAX_MS</code>	Number	30000	The longest, in milliseconds, n8n waits between recovery attempts.
<code>DB_CONNECTION_ACQUISITION_TIMEOUT_MS</code>	Number	30000	How long, in milliseconds, a query waits while recovery is in progress before failing fast with an error.
<code>DB_POSTGRESDB_DATABASE</code> <code>/_FILE</code>	String	n8n	The name of the PostgreSQL database.
<code>DB_POSTGRESDB_HOST</code> <code>/_FILE</code>	String	localhost	The PostgreSQL host.
<code>DB_POSTGRESDB_PORT</code> <code>/_FILE</code>	Number	5432	The PostgreSQL port.
<code>DB_POSTGRESDB_USER</code> <code>/_FILE</code>	String	postgres	The PostgreSQL user.
<code>DB_POSTGRESDB_PASSWORD</code> <code>/_FILE</code>	String	-	The PostgreSQL password.
<code>DB_POSTGRESDB_POOL_SIZE</code> <code>/_FILE</code>	Number	2	Control how many parallel open Postgres connections n8n should have.
<code>DB_POSTGRESDB_MAX_CONNECTION_LIFETIME_MS</code> <code>/_FILE</code>	Number	3600000	How long, in milliseconds, n8n keeps a pooled connection before recycling it.
<code>DB_POSTGRESDB_KEEP_ALIVE</code> <code>/_FILE</code>	Boolean	true	Whether to enable TCP keep-alive on connections.
<code>DB_POSTGRESDB_KEEP_ALIVE_INITIAL_DELAY_MS</code> <code>/_FILE</code>	Number	10000	How long, in milliseconds, n8n waits before sending the first TCP keep-alive probe on a connection.
<code>DB_POSTGRESDB_SCHEMA</code> <code>/_FILE</code>	String	public	The PostgreSQL schema.
<code>DB_POSTGRESDB_SSL_ENABLED</code> <code>/_FILE</code>	Boolean	false	Whether to enable SSL. Automatically enabled if <code>DB_POSTGRESDB_SSL_CA</code> , <code>DB_POSTGRESDB_SSL_CERT</code> or <code>DB_POSTGRESDB_SSL_KEY</code> is...
<code>DB_POSTGRESDB_SSL_CA</code> <code>/_FILE</code>	String	-	The PostgreSQL SSL certificate authority.
<code>DB_POSTGRESDB_SSL_CERT</code> <code>/_FILE</code>	String	-	The PostgreSQL SSL certificate.

<code>DB_POSTGRESDB_SSL_KEY</code> <code>/_FILE</code>	String	-	The PostgreSQL SSL key.
<code>DB_POSTGRESDB_SSL_REJECT_UNAUTHORIZED</code> <code>/_FILE</code>	Boolean	<code>true</code>	If n8n should reject unauthorized SSL connections (true) or not (false).
<code>DB_SQLITE_POOL_SIZE</code>	Number	<code>0</code>	Controls whether to open the SQLite file in WAL mode or rollback journal mode.
<code>DB_SQLITE_VACUUM_ON_STARTUP</code>	Boolean	<code>false</code>	Runs VACUUM operation on startup to rebuild the database.
:::			

33.7 Nodes

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>N8N_COMMUNITY_PACKAGES_AUTH_TOKEN</code>	String	-	Authentication token for a private npm registry.
<code>N8N_COMMUNITY_PACKAGES_ENABLED</code>	Boolean	<code>true</code>	Enables (true) or disables (false) the functionality to install and load community nodes.
<code>N8N_COMMUNITY_PACKAGES_PREVENT_LOADING</code>	Boolean	<code>false</code>	Prevents (true) or allows (false) loading installed community nodes on instance startup.
<code>N8N_COMMUNITY_PACKAGES_REGISTRY</code>	String	<code>'https://registry.npmjs.o...</code>	NPM registry URL to pull community packages from (license required).
<code>N8N_CUSTOM_EXTENSIONS</code>	String	-	Specify the path to directories containing your custom nodes.
<code>N8N_PYTHON_ENABLED</code>	Boolean	<code>true</code>	Whether to enable Python execution on the Code node.
<code>N8N_UNVERIFIED_PACKAGES_ENABLED</code>	Boolean	<code>true</code>	When <code>N8N_COMMUNITY_PACKAGES_ENABLED</code> is true, this variable controls whether to enable the installation and use of unverified...
<code>N8N_VERIFIED_PACKAGES_ENABLED</code>	Boolean	<code>true</code>	When <code>N8N_COMMUNITY_PACKAGES_ENABLED</code> is true, this variable controls whether to show verified community nodes in the nodes...
<code>NODE_FUNCTION_ALLOW_BUILTIN</code>	String	-	Permit users to import specific built-in modules in the Code node.

<code>NODE_FUNCTION_ALLOW_EXTERNAL</code>	String	-	Permit users to import specific external modules (from <code>n8n/node_modules</code>) in the Code node. n8n disables importing modules by...
<code>NODES_ERROR_TRIGGER_TYPE</code>	String	<code>`n8n-nodes-base.errorTrig...</code>	Specify which node type to use as Error Trigger.
<code>NODES_EXCLUDE</code>	Array of strings	<code>`["n8n-nodes-base.execut...</code>	Specify which nodes not to load.
<code>NODES_INCLUDE</code>	Array of strings	-	Specify which nodes to load.
<code>NODES_MERGE_SQL_SANDBOX_MEMORY_LIMIT_MB</code>	Number	<code>64</code>	Memory limit in MB for the Merge node's SQL sandbox.
<code>N8N_COMPRESSION_NODE_MAX_DECOMPRESSED_SIZE_BYTES</code>	Number	<code>268435456</code>	Maximum total decompressed output size in bytes.
<code>N8N_COMPRESSION_NODE_MAX_ZIP_ENTRIES</code>	Number	<code>1000</code>	Maximum number of entries allowed in a ZIP archive.
...			

33.8 Logs

compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>N8N_LOG_LEVEL</code>	Enum string: <code>info</code> , <code>warn</code> , <code>error</code> , <code>debug</code>	<code>info</code>	Log output level.
<code>N8N_LOG_OUTPUT</code>	Enum string: <code>console</code> , <code>file</code>	<code>console</code>	Where to output logs.
<code>N8N_LOG_FORMAT</code>	Enum string: <code>text</code> , <code>json</code>	<code>text</code>	The log format to use. <code>text</code> prints human readable messages. <code>json</code> prints one JSON object per line containing the message,...
<code>N8N_LOG_CRON_ACTIVE_INTERVAL</code>	Number	<code>0</code>	Interval in minutes to log currently active cron jobs.
<code>N8N_LOG_FILE_COUNT_MAX</code>	Number	<code>100</code>	Max number of log files to keep.
<code>N8N_LOG_FILE_SIZE_MAX</code>	Number	<code>16</code>	Max size of each log file in MB.
<code>N8N_LOG_FILE_LOCATION</code>	String	<code>`/log...</code>	Log file location.
<code>DB_LOGGING_ENABLED</code>	Boolean	<code>false</code>	Whether to enable database-specific logging.

<code>DB_LOGGING_OPTIONS</code>	Enum string: <code>query</code> , <code>error</code> , <code>schema</code> , <code>warn</code> , <code>info</code> , <code>log</code>	<code>error</code>	Database log output level.
<code>DB_LOGGING_MAX_EXECUTION_TIME</code>	Number	<code>1000</code>	Maximum execution time (in milliseconds) before n8n logs a warning.
<code>CODE_ENABLE_STDOUT</code>	Boolean	<code>false</code>	Set to <code>true</code> to send Code node logs from <code>console.log</code> or <code>print</code> to the process's stdout, only for production executions.
<code>NO_COLOR</code>	any	<code>undefined</code>	Set to any value to output logs without ANSI colors.
<code>N8N_EVENTBUS_CHECKUNSENTINTERVAL</code>	Number	<code>0</code>	How often (in milliseconds) to check for unsent event messages.
<code>N8N_EVENTBUS_LOGWRITER_SYNCFILEACCESS</code>	Boolean	<code>false</code>	Whether all file access happens synchronously within the thread (true) or not (false).
<code>N8N_EVENTBUS_LOGWRITER_KEEPLLOGCOUNT</code>	Number	<code>3</code>	Number of event log files to keep.
<code>N8N_EVENTBUS_LOGWRITER_MAXFILESIZEINKB</code>	Number	<code>10240</code>	Maximum size (in kilobytes) of an event log file before a new one starts.
<code>N8N_EVENTBUS_LOGWRITER_LOGBASENAME</code>	String	<code>n8nEventLog</code>	Basename of the event log file.
<code>N8N_EVENTBUS_LOGWRITER_LOGFULLPATH</code>	String	<code>''</code>	Absolute path to the event log file.
<code>N8N_EVENTBUS_LOGWRITER_MAXTOTALMESSAGESPERFILE</code>	Number	<code>500000</code>	Maximum number of lines parsed from a single event log file during recovery.
...			

33.9 Timezone and Localization

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
GENERIC_TIMEZONE	*	America/New_York	The n8n instance timezone.
N8N_DEFAULT_LOCALE	String	en	A locale identifier, compatible with the Accept-Language header. n8n doesn't support regional identifiers, such as de-AT.

:::

33.10 Workflows

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
N8N_ONBOARDING_FLOW_DISABLED	Boolean	false	Whether to disable onboarding tips when creating a new workflow (true) or not (false).
N8N_WORKFLOW_CALLER_POLICY_DEFAULT_OPTION	String	workflowsFromSameOwner	Which workflows can call a workflow.
N8N_WORKFLOW_TAGS_DISABLED	Boolean	false	Whether to disable workflow tags (true) or enable tags (false).
WORKFLOWS_DEFAULT_NAME	String	My workflow	The default name used for new workflows.

:::

33.11 Credentials

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>CREDENTIALS_OVERWRITE_DATA</code> <code>/_FILE</code>	*	-	Overwrites for credentials.
<code>CREDENTIALS_OVERWRITE_ENDPOINT</code>	String	-	The API endpoint to fetch credentials.
<code>CREDENTIALS_OVERWRITE_PERSISTENCE</code>	Boolean	<code>false</code>	Enable database persistence for credential overwrites.
<code>N8N_MANAGED_OAUTH_SHOW_SCOPES</code>	String	-	Comma-separated list of managed OAuth credential types for which users can configure scope fields.
<code>CREDENTIALS_DEFAULT_NAME</code>	String	<code>My credentials</code>	The default name for credentials.
...			

33.12 Binary Data

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>N8N_AVAILABLE_BINARY_DATA_MODES</code>	String	<code>filesystem</code>	A comma separated list of available binary data modes.
<code>N8N_BINARY_DATA_DATABASE_MAX_FILE_SIZE</code>	Number	<code>512</code>	Maximum size (in MiB) of a single file n8n stores when <code>N8N_DEFAULT_BINARY_DATA_MODE</code> is <code>database</code> .
<code>N8N_BINARY_DATA_STORAGE_PATH</code>	String	<code>`N8N_USER_FOLDER/binaryDa...</code>	The path where n8n stores binary data.
<code>N8N_DEFAULT_BINARY_DATA_MODE</code>	String	<code>default</code>	The default binary data mode. <code>default</code> keeps binary data in memory.
...			

33.13 Queue Mode

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
OFFLOAD_MANUAL_EXECUTIONS_TO_WORKERS	Boolean	false	Set to <code>true</code> if you want manual executions to run on the worker rather than on main.
QUEUE_BULL_PREFIX	String	-	Prefix to use for all queue keys.
QUEUE_BULL_REDIS_DB	Number	0	The Redis database used.
QUEUE_BULL_REDIS_HOST	String	localhost	The Redis host.
QUEUE_BULL_REDIS_PORT	Number	6379	The Redis port used.
QUEUE_BULL_REDIS_USERNAME	String	-	The Redis username (needs Redis version 6 or above).
QUEUE_BULL_REDIS_PASSWORD	String	-	The Redis password.
QUEUE_BULL_REDIS_TIMEOUT_THRESHOLD	Number	10000	The Redis timeout threshold (in ms).
QUEUE_BULL_REDIS_CLUSTER_NODES	String	-	Expects a comma-separated list of Redis Cluster nodes in the format <code>host:port</code> , for the Redis client to initially connect to.
QUEUE_BULL_REDIS_TLS	Boolean	false	Enable TLS on Redis connections.
QUEUE_BULL_REDIS_DUALSTACK	Boolean	false	Enable dual-stack support (IPv4 and IPv6) on Redis connections.
QUEUE_WORKER_TIMEOUT (deprecated)	Number	30	Deprecated from n8n 1.22.0. Use <code>N8N_GRACEFUL_SHUTDOWN_TIMEOUT</code> instead.
QUEUE_HEALTH_CHECK_ACTIVE	Boolean	false	Whether to enable health checks (true) or disable (false).
QUEUE_HEALTH_CHECK_PORT	Number	5678	The port to serve health checks on.
QUEUE_WORKER_LOCK_DURATION	Number	60000	How long (in ms) is the lease period for a worker to work on a message.
QUEUE_WORKER_LOCK_RENEW_TIME	Number	10000	How frequently (in ms) should a worker renew the lease time.
QUEUE_WORKER_STALLED_INTERVAL	Number	30000	How often should a worker check for stalled jobs (use 0 for never).
QUEUE_WORKER_MAX_STALLED_COUNT (deprecated)	Number	1	Deprecated Removed in n8n 2.0. Setting this has no effect.

<code>N8N_WEBHOOK_RESPONSE_RELAY_SIZE_MAX</code>	Number	<code>64</code>	Maximum size (in MiB) of a response a worker sends back to the main instance inside a queue message.
<code>N8N_WEBHOOK_RESPONSE_RELAY_OFFLOAD_ENABLED</code>	Boolean	<code>false</code>	Whether a worker stores a response body above <code>N8N_WEBHOOK_RESPONSE_RELAY_SIZE_MAX</code> in binary data storage, so the main...
<code>N8N_MULTI_MAIN_SETUP_ENABLED</code>	Boolean	<code>false</code>	Whether to enable multi-main setup for queue mode (license required).
<code>N8N_MULTI_MAIN_SETUP_KEY_TTL</code>	Number	<code>10</code>	Time to live (in seconds) for leader key in multi-main setup.
<code>N8N_MULTI_MAIN_SETUP_CHECK_INTERVAL</code>	Number	<code>3</code>	Interval (in seconds) for leader check in multi-main setup.
...			

33.14 Task Runners

:::compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>N8N_RUNNERS_ENABLED</code> (deprecated)	Boolean	<code>false</code>	Are task runners enabled. Deprecated from n8n 2.0.
<code>N8N_RUNNERS_MODE</code>	Enum string: <code>internal</code> (deprecated), <code>external</code>	<code>internal</code>	How to launch and run the task runner. <code>internal</code> means n8n will launch a task runner as child process. Deprecated: ...
<code>N8N_RUNNERS_AUTH_TOKEN</code>	String	Random string	Shared secret used by a task runner to authenticate to n8n.
<code>N8N_RUNNERS_BROKER_PORT</code>	Number	<code>5679</code>	Port the task broker listens on for task runner connections.
<code>N8N_RUNNERS_BROKER_LISTEN_ADDRESS</code>	String	<code>127.0.0.1</code>	Address the task broker listens on.
<code>N8N_RUNNERS_MAX_PAYLOAD</code>	Number	<code>1 073 741 824</code>	Maximum payload size in bytes for communication between a task broker and a task runner.
<code>N8N_RUNNERS_MAX_OLD_SPACE_SIZE</code>	String		The <code>--max-old-space-size</code> option to use for a task runner (in MB).
<code>N8N_RUNNERS_MAX_CONCURRENCY</code>	Number	<code>5</code>	The number of concurrent tasks a task runner can execute at a time.
<code>N8N_RUNNERS_TASK_TIMEOUT</code>	Number	<code>300</code>	The maximum time, in seconds, a task can run before the runner stops it and restarts.

<code>N8N_RUNNERS_HEARTBEAT_INTERVAL</code>	Number	<code>30</code>	The interval, in seconds, at which the runner must send a heartbeat to the broker.
<code>N8N_RUNNERS_INSECURE_MODE</code>	Boolean	<code>false</code>	Whether to disable all security measures in the task runner, for compatibility with modules that rely on insecure JS features....
<code>N8N_RUNNERS_TASK_REQUEST_TIMEOUT</code>	Number	<code>60</code>	How long (in seconds) a task request can wait for a runner to become available before timing out.
<code>N8N_RUNNERS_LAUNCHER_LOG_LEVEL</code>	Enum string: <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code>	<code>info</code>	Which log messages to show.
<code>N8N_RUNNERS_AUTH_TOKEN</code>	String	-	Shared secret used to authenticate to n8n.
<code>N8N_RUNNERS_AUTO_SHUTDOWN_TIMEOUT</code>	Number	<code>15</code>	The number of seconds to wait before shutting down an idle runner.
<code>N8N_RUNNERS_TASK_BROKER_URI</code>	String	<code>http://127.0.0.1:5679</code>	The URI of the task broker server (n8n instance).
<code>N8N_RUNNERS_LAUNCHER_HEALTH_CHECK_PORT</code>	Number	<code>5680</code>	Port for the launcher's health check server.
<code>N8N_RUNNERS_MAX_PAYLOAD</code>	Number	<code>1 073 741 824</code>	Maximum payload size in bytes for communication between a task broker and a task runner.
<code>N8N_RUNNERS_MAX_CONCURRENCY</code>	Number	<code>5</code>	The number of concurrent tasks a task runner can execute at a time.
<code>N8N_RUNNERS_GRANT_TOKEN</code>	String	Random string	Token the runner uses to authenticate with the task broker.
<code>N8N_RUNNERS_ID</code>	String	Random string	ID the runner identifies itself with to the task broker.

<code>N8N_RUNNERS_AUTO_SHUTDOWN_TIMEOUT</code>	Number	<code>15</code>	The number of seconds to wait before shutting down an idle runner.
<code>N8N_RUNNERS_TASK_BROKER_URI</code>	String	<code>http://127.0.0.1:5679</code>	The URI of the task broker server (n8n instance).
<code>N8N_RUNNERS_LAUNCHER_HEALTH_CHECK_PORT</code>	Number	<code>5680</code>	Port for the launcher's health check server.
<code>N8N_RUNNERS_MAX_PAYLOAD</code>	Number	<code>1 073 741 824</code>	Maximum payload size in bytes for communication between a task broker and a task runner.
<code>N8N_RUNNERS_MAX_CONCURRENCY</code>	Number	<code>5</code>	The number of concurrent tasks a task runner can execute at a time.
<code>NODE_FUNCTION_ALLOW_BUILTIN</code>	String	-	Permit users to import specific built-in modules in the Code node.
<code>NODE_FUNCTION_ALLOW_EXTERNAL</code>	String	-	Permit users to import specific external modules (from <code>n8n/node_modules</code>) in the Code node. n8n disables importing modules by...
<code>N8N_RUNNERS_ALLOW_PROTOTYPE_MUTATION</code>	Boolean	<code>false</code>	Whether to allow prototype mutation for external libraries.
<code>GENERIC_TIMEZONE</code>	*	<code>America/New_York</code>	The same default timezone as configured for the n8n instance.
<code>NODE_OPTIONS</code>	String	-	Options for Node.js.
<code>N8N_RUNNERS_MAX_OLD_SPACE_SIZE</code>	String		The <code>--max-old-space-size</code> option to use for a task runner (in MB).
<code>N8N_RUNNERS_STDLIB_ALLOW</code>	String	-	Python standard library modules that you can use in the Code node, including their submodules.
<code>N8N_RUNNERS_EXTERNAL_ALLOW</code>	String	-	Third-party Python modules that are allowed to be used in the Code node, including their submodules.
<code>N8N_RUNNERS_ALLOW_TRANSITIVE_IMPORTS</code>	Boolean	<code>false</code>	Whether imports performed by an allowlisted package's own code skip the import allowlist (both <code>N8N_RUNNERS_EXTERNAL_ALLOW</code> ...

<code>N8N_RUNNERS_BUILTINS_DENY</code>	String	<code>`eval,exec,compile,open,i...</code>	Python built-ins that you can't use in the Code node.
<code>N8N_BLOCK_RUNNER_ENV_ACCESS</code>	Boolean	<code>true</code>	Whether to block access to the runner's environment from within Python code tasks.
...			

33.15 User Management and 2FA

compact

VARIABLE	TYPE	DEFAULT	MEANING
<code>N8N_EMAIL_MODE</code>	String	<code>smtp</code>	Enable emails.
<code>N8N_SMTP_HOST</code>	String	-	<i>your_SMTP_server_name.</i>
<code>N8N_SMTP_PORT</code>	Number	-	<i>your_SMTP_server_port.</i>
<code>N8N_SMTP_USER</code>	String	-	<i>your_SMTP_username.</i>
<code>N8N_SMTP_PASS</code>	String	-	<i>your_SMTP_password.</i>
<code>N8N_SMTP_OAUTH_SERVICE_CLIENT</code>	String	-	If using 2LO with a service account this is your client ID.
<code>N8N_SMTP_OAUTH_PRIVATE_KEY</code>	String	-	If using 2LO with a service account this is your private key.
<code>N8N_SMTP_SENDER</code>	String	-	Sender email address.
<code>N8N_SMTP_SSL</code>	Boolean	<code>true</code>	Whether to use SSL for SMTP (true) or not (false).
<code>N8N_SMTP_STARTTLS</code>	Boolean	<code>true</code>	Whether to use STARTTLS for SMTP (true) or not (false).
<code>N8N_UM_EMAIL_TEMPLATES_INVITE</code>	String	-	Full path to your HTML email template.
<code>N8N_UM_EMAIL_TEMPLATES_PWRESET</code>	String	-	Full path to your HTML email template.
<code>N8N_UM_EMAIL_TEMPLATES_WORKFLOW_SHARED</code>	String	-	Overrides the default HTML template for notifying users that a workflow was shared.
<code>N8N_UM_EMAIL_TEMPLATES_CREDENTIALS_SHARED</code>	String	-	Overrides the default HTML template for notifying users that a credential was shared.
<code>N8N_UM_EMAIL_TEMPLATES_PROJECT_SHARED</code>	String	-	Overrides the default HTML template for notifying users that a project was shared.
<code>N8N_USER_MANAGEMENT_JWT_SECRET</code>	String	-	Set a specific JWT secret.

N8N_USER_MANAGEMENT_JWT_DURATION_HOURS	Number	168	Set an expiration date for the JWTs in hours.
N8N_USER_MANAGEMENT_JWT_REFRESH_TIMEOUT_HOURS	Number	0	How many hours before the JWT expires to automatically refresh it. 0 means 25% of N8N_USER_MANAGEMENT_JWT_DURATION_HOURS . -1...
N8N_MFA_ENABLED	Boolean	true	Whether to enable two-factor authentication (true) or disable (false). n8n ignores this if existing users have 2FA enabled.
N8N_INVITE_LINKS_EMAIL_ONLY	Boolean	false	When set to true, n8n will only deliver invite links via email and will not expose them through the API. This option enhances...
:::			

Built-In Node Directory

Every core node, trigger node, and AI root node documented in the official n8n docs, A–Z. Generated from the n8n documentation source (github.com/n8n-io/n8n-docs) in October 2026. n8n also has hundreds of **app nodes** (Google Sheets, Slack, Notion...) for actions inside specific services. Search the nodes panel for those.

summary In this chapter

- Core nodes: the building blocks
- Trigger nodes: every documented way a workflow can start from an outside app
- AI root nodes: agents, chains, and vector stores

TIP

In the editor, press **Tab** or click + and type part of a node name. The names below are the ones you'll see in that panel.

compact

NODE	KIND	WHAT IT DOES
Activation Trigger	Core	The Activation Trigger node gets triggered when an event gets fired by n8n or a workflow.
ActiveCampaign Trigger	Trigger	N8n provides an app node for ActiveCampaign.
Acuity Scheduling Trigger	Trigger	Starts a workflow when an event happens in Acuity Scheduling.
Affinity Trigger	Trigger	N8n provides an app node for Affinity.
Aggregate	Core	Take separate items, or portions of them, and group them together into individual items.

AI Agent	AI (cluster root)	The AI Agent node lets you build an AI agent in n8n.
AI Transform	Core	Generate code snippets based on your prompt.
Airtable Trigger	Trigger	N8n provides an app node for Airtable.
AMQP Trigger	Trigger	Starts a workflow when an event happens in AMQP.
Asana Trigger	Trigger	N8n provides an app node for Asana.
Autopilot Trigger	Trigger	Starts a workflow when an event happens in Autopilot.
AWS SNS Trigger	Trigger	Starts a workflow when a message is published to an AWS SNS topic.
Azure AI Search Vector Store	AI (cluster root)	Azure AI Search (formerly Azure Cognitive Search) is a cloud search service with vector search capabilities for RAG and semantic search applications.
Basic LLM Chain	AI (cluster root)	Set the prompt that the model will use along with setting an optional parser for the response.
Bitbucket Trigger	Trigger	Starts a workflow when an event happens in Bitbucket.
Box Trigger	Trigger	To get your Target ID in Box:.
Brevo Trigger	Trigger	N8n provides an app node for Brevo.
Cal Trigger	Trigger	Starts a workflow when an event happens in Cal.
Calendly Trigger	Trigger	Here are some common errors and issues with the Calendly Trigger node and steps to resolve or troubleshoot them.
Chargebee Trigger	Trigger	To add a Webhook URL in Chargebee:.
Chat	Core	Send messages into the chat and optionally wait for responses from users.
Chat Trigger	Core	Use the Chat Trigger node when building AI workflows for chatbots and other chat interfaces.
Chroma Vector Store	AI (cluster root)	Interact with your Chroma database as vector store.
ClickUp Trigger	Trigger	N8n provides an app node for ClickUp.
Clockify Trigger	Trigger	This node uses the workflow timezone setting to specify the range of time entries starting time.
Code	Core	The Code node lets you run your own JavaScript or Python inside a workflow, so you can transform data or add logic the built-in nodes don't cover.
Compare Datasets	Core	The Compare Datasets node helps you compare data from two input streams.
Compression	Core	Compress and decompress files.
Convert to File	Core	Take input data and output it as a file.
ConvertKit Trigger	Trigger	N8n provides an app node for ConvertKit.
Copper Trigger	Trigger	N8n provides an app node for Copper.

Crypto	Core	Perform cryptographic operations in workflows.
Customer.io Trigger	Trigger	N8n provides an app node for Customer.io.
Data table	Core	Create and manage internal data tables.
Databricks Trigger	Trigger	Respond to events in Databricks and integrate Databricks with other applications.
Date & Time	Core	The Date & Time node manipulates date and time data and convert it to different formats.
Debug Helper	Core	Trigger different error types or generate random datasets to help test n8n workflows.
Edit Fields (Set)	Core	Set workflow data.
Edit Image	Core	Manipulate and edit images.
Email Trigger (IMAP)	Core	Receive emails using an IMAP email server.
Emelia Trigger	Trigger	Starts a workflow when an event happens in Emelia.
Error Trigger	Core	You can use the Error Trigger node to create error workflows.
Evaluation	Core	The Evaluation node performs various operations related to evaluations to validate your AI workflow reliability.
Evaluation Trigger	Core	Use the Evaluation Trigger node when setting up evaluations to validate your AI workflow reliability.
Eventbrite Trigger	Trigger	Starts a workflow when an event happens in Eventbrite.
Execute Command	Core	The Execute Command node runs shell commands on the host machine that runs n8n.
Execute Sub-workflow	Core	Run a different workflow on the host machine that runs n8n.
Execute Sub-workflow Trigger	Core	Use this node to start a workflow in response to another workflow.
Execution Data	Core	Use this node to save metadata for workflow executions.
Extract From File	Core	A common pattern in n8n workflows is to receive a file, either from an [HTTP Request node][] (for files you are fetching from a website), a...
Facebook Lead Ads Trigger	Trigger	Respond to events in Facebook Lead Ads and integrate Facebook Lead Ads with other applications. n8n has built-in support for responding to new leads.
Facebook Trigger	Trigger	Trigger a workflow when events occur in Facebook.
Figma Trigger (Beta)	Trigger	Starts a workflow when an event happens in Figma.
Filter	Core	Filter items based on a condition.
Flow Trigger	Trigger	Flow is modern task and project management software for teams.
Form.io Trigger	Trigger	Starts a workflow when an event happens in Form.io.

Formstack Trigger	Trigger	Starts a workflow when an event happens in Formstack.
FTP	Core	The FTP node is useful to access and upload files to an FTP or SFTP server.
GetResponse Trigger	Trigger	Starts a workflow when an event happens in GetResponse.
Git	Core	Configure this operation with these parameters:.
GitHub Trigger	Trigger	N8n provides an app node for GitHub.
GitLab Trigger	Trigger	If your repository is located within nested GitLab groups or subgroups (for example, <code>https://gitlab.com/org/subgroup1/subgroup2/my-repo</code>),...
Gmail Trigger	Trigger	Configure the node with these parameters:.
Google Business Profile Trigger	Trigger	Respond to events in Google Business Profile and integrate Google Business Profile with other applications. n8n has built-in support for...
Google Calendar Trigger	Trigger	N8n provides an app node for Google Calendar.
Google Drive Trigger	Trigger	For common questions or issues and suggested solutions, refer to Common issues.
Google Sheets Trigger	Trigger	N8n provides an app node for Google Sheets.
GraphQL	Core	Select the type of authentication to use. n8n recommends using Predefined Credential Type when it's available for the service you want to...
Guardrails	Core	Enforce safety, security, and content policies on text.
Gumroad Trigger	Trigger	Starts a workflow when an event happens in Gumroad.
Help Scout Trigger	Trigger	Starts a workflow when an event happens in Help Scout.
HTML	Core	The HTML node provides operations to help you work with HTML in n8n.
HTTP Request	Core	The HTTP Request node is one of the most versatile nodes in n8n.
HubSpot Trigger	Trigger	N8n provides an app node for HubSpot.
If	Core	Split a workflow conditionally based on comparison operations.
Information Extractor	AI (cluster root)	Extract structured information from incoming data.
Invoice Ninja Trigger	Trigger	Starts a workflow when an event happens in Invoice Ninja.
Jira Trigger	Trigger	Starts a workflow when an event happens in Jira.
Jotform Trigger	Trigger	Starts a workflow when an event happens in Jotform.
JWT	Core	Work with JSON web tokens in your n8n workflows.
Kafka Trigger	Trigger	Version 2 runs on a new, actively maintained Kafka engine.

Keap Trigger	Trigger	Starts a workflow when an event happens in Keap.
KoboToolbox Trigger	Trigger	This node starts a workflow upon new submissions of a specified form.
LangChain Code	AI (cluster root)	Import LangChain.
LDAP	Core	This node allows you to interact with your LDAP servers to create, find, and update objects.
Lemlist Trigger	Trigger	Starts a workflow when an event happens in Lemlist.
Limit	Core	Remove items beyond a defined maximum number.
Linear Trigger	Trigger	Starts a workflow when an event happens in Linear.
Local File Trigger	Core	The Local File Trigger node starts a workflow when it detects changes on the file system.
LoneScale Trigger	Trigger	Respond to workflow events in LoneScale and integrate LoneScale with other applications.
Loop Over Items	Core	The Loop Over Items node helps you loop through data when needed.
Mailchimp Trigger	Trigger	Starts a workflow when an event happens in Mailchimp.
MailerLite Trigger	Trigger	Starts a workflow when an event happens in MailerLite.
Mailjet Trigger	Trigger	Starts a workflow when an event happens in Mailjet.
Manual Trigger	Core	Use this node if you want to start a workflow by selecting Execute Workflow and don't want any option for the workflow to run automatically.
Markdown	Core	The Markdown node converts between Markdown and HTML formats.
Mautic Trigger	Trigger	N8n provides an app node for Mautic.
MCP Server Trigger	Core	Allow n8n to act as a Model Context Protocol (MCP) server, making n8n tools and workflows available to MCP clients.
Merge	Core	Combine data from multiple streams, once data of all streams is available.
Message an Agent	Core	Send a message to an agent from a workflow, then use the agent's reply in the nodes that follow.
Microsoft Agent 365 Trigger	AI (cluster root)	Receive messages from Microsoft Agent 365 and respond with AI-powered agent capabilities.
Microsoft OneDrive Trigger	Trigger	Respond to events in Microsoft OneDrive and integrate Microsoft OneDrive with other applications. n8n has built-in support for file and folder...
Microsoft Outlook Trigger	Trigger	Respond to events in Microsoft Outlook and integrate Microsoft Outlook with other applications.
Microsoft SharePoint Trigger	Trigger	Start a workflow when a file or a list item changes in Microsoft SharePoint.

Microsoft Teams Trigger	Trigger	Respond to events in Microsoft Teams and integrate Microsoft Teams with other applications.
Milvus Vector Store	AI (cluster root)	Interact with your Milvus database as vector store.
MongoDB Atlas Vector Store	AI (cluster root)	MongoDB Atlas Vector Search is a feature of MongoDB Atlas that enables users to store and query vector embeddings.
MQTT Trigger	Trigger	Starts a workflow when an event happens in MQTT.
n8n	Core	A node to integrate with n8n itself.
n8n Form	Core	Create user-facing forms with multiple steps.
n8n Form Trigger	Core	Use the n8n Form trigger to start a workflow when a user submits a form, taking the input data from the form.
n8n Trigger	Core	The n8n Trigger node triggers when the workflow containing this node updates or gets published, or when the n8n instance starts or restarts.
Netlify Trigger	Trigger	N8n provides an app node for Netlify.
No Operation, do nothing	Core	Use the No Operation, do nothing node when you don't want to perform any operations.
Notion Trigger	Trigger	N8n provides an app node for Notion.
Onfleet Trigger	Trigger	Trigger a workflow on:.
Oracle Database Vector Store	AI (cluster root)	Interact with Oracle Database as a vector store.
PayPal Trigger	Trigger	Starts a workflow when an event happens in PayPal.
PGVector Vector Store	AI (cluster root)	PGVector is an extension of PostgreSQL.
Pinecone Vector Store	AI (cluster root)	Interact with your Pinecone database as vector store.
Pipedrive Trigger	Trigger	Starts a workflow when an event happens in Pipedrive.
Postgres Trigger	Trigger	Respond to events in Postgres and integrate Postgres with other applications. n8n has built-in support responding to insert, update, and delete...
Postmark Trigger	Trigger	Starts a workflow when an event happens in Postmark.
Pushcut Trigger	Trigger	Follow these steps to configure your Pushcut Trigger node with your Pushcut app.
Qdrant Vector Store	AI (cluster root)	Interact with your Qdrant collection as a vector store.
Question and Answer Chain	AI (cluster root)	Use a vector store as a retriever.
RabbitMQ Trigger	Trigger	Starts a workflow when a message arrives in a RabbitMQ queue.

Read/Write Files from Disk	Core	Read and write files from/to the machine where n8n is running.
Redis Trigger	Trigger	Subscribe to a Redis channel.
Redis Vector Store	AI (cluster root)	Interact with your Redis database as a vector store.
Remove Duplicates	Core	Identify and delete items that are:.
Rename Keys	Core	Rename the keys of a key-value pair in n8n.
Respond to Webhook	Core	Control the response to incoming webhooks.
RSS Feed Trigger	Core	The RSS Feed Trigger node allows you to start an n8n workflow when a new RSS feed item has been published.
RSS Read	Core	Read data from RSS feeds published on the internet.
Salesforce Trigger	Trigger	Respond to events in Salesforce and integrate Salesforce with other applications.
Schedule Trigger	Core	Run workflows at fixed intervals and times.
SeaTable Trigger	Trigger	Starts a workflow when an event happens in SeaTable.
Send Email	Core	The Send Email node sends emails using an SMTP email server.
Sentiment Analysis	AI (cluster root)	Analyze the sentiment of incoming text data.
Shopify Trigger	Trigger	Starts a workflow when an event happens in Shopify.
Simple Vector Store	AI (cluster root)	Store and retrieve embeddings in n8n's in-app memory.
Slack Trigger	Trigger	Respond to events in Slack and integrate Slack with other applications.
Sort	Core	Organize lists of items in a desired ordering, or generate a random selection.
Split Out	Core	Separate a single data item containing a list into multiple items.
SSE Trigger	Core	Server-Sent Events (SSE) is a server push technology enabling a client to receive automatic updates from a server using HTTP connection.
SSH	Core	The SSH node is useful for executing commands using the Secure Shell Protocol.
Stop And Error	Core	Display custom error messages, cause executions to fail under certain conditions, and send custom error information to error workflows.
Strava Trigger	Trigger	Starts a workflow when an event happens in Strava.
Stripe Trigger	Trigger	The Stripe Trigger node can verify that incoming webhook requests genuinely come from Stripe.

Summarization Chain	AI (cluster root)	Summarize multiple documents.
Summarize	Core	Aggregate items together, in a manner similar to Excel pivot tables.
Supabase Vector Store	AI (cluster root)	Use the Supabase Vector Store to interact with your Supabase database as vector store.
SurveyMonkey Trigger	Trigger	Starts a workflow when an event happens in SurveyMonkey.
Switch	Core	Route a workflow conditionally based on comparison operations.
Taiga Trigger	Trigger	Starts a workflow when an event happens in Taiga.
Telegram Trigger	Trigger	Some events may require additional permissions , see Telegram's API documentation for more information.
Text Classifier	AI (cluster root)	Classify (categorize) incoming data.
TheHive 5 Trigger	Trigger	Respond to events in TheHive and integrate TheHive with other applications.
TheHive Trigger	Trigger	N8n provides an app node for TheHive.
Toggl Trigger	Trigger	Starts a workflow when an event happens in Toggl.
TOTP	Core	The TOTP node provides a way to generate a TOTP (time-based one-time password).
Track Time Saved	Core	Record how many minutes a workflow saves, based on the path an execution actually takes. n8n adds up every Track Time Saved node that runs during...
Trello Trigger	Trigger	The model ID is the ID of any model in Trello.
Twilio Trigger	Trigger	Respond to events in Twilio and integrate Twilio with other applications.
Typeform Trigger	Trigger	Starts a workflow when an event happens in Typeform.
Venafi TLS Protect Cloud Trigger	Trigger	Start a workflow in n8n in response to events in the cloud-based Venafi TLS Protect service.
Wait	Core	Use the Wait node pause your workflow's execution.
Weaviate Vector Store	AI (cluster root)	Interact with your Weaviate collection as a vector store.
Webex by Cisco Trigger	Trigger	Starts a workflow when an event happens in Webex by Cisco.
Webflow Trigger	Trigger	Starts a workflow when an event happens in Webflow.
Webhook	Core	Create webhooks, which can receive data from apps and services when an event occurs.

WhatsApp Trigger	Trigger	Respond to events in WhatsApp and integrate WhatsApp with other applications.
Wise Trigger	Trigger	Starts a workflow when an event happens in Wise.
WooCommerce Trigger	Trigger	Starts a workflow when an event happens in WooCommerce.
Workable Trigger	Trigger	Respond to events in the Workable recruiting platform and integrate Workable with other applications.
Workflow Trigger	Core	The Workflow Trigger node gets triggered when a workflow is updated or activated.
Wufoo Trigger	Trigger	Starts a workflow when an event happens in Wufoo.
XML	Core	Convert data from and to XML.
Zendesk Trigger	Trigger	Starts a workflow when an event happens in Zendesk.
Zep Vector Store	AI (cluster root)	Use the Zep Vector Store to interact with Zep vector databases.
⋮		

Quick Reference

:::cheatsheet **Pattern:** Trigger → get data → reshape → decide → act

Start: n8n Cloud trial (14 days) · or `curl -fsSL https://get.n8n.io | sh` · or Docker `n8nio/n8n` → <http://localhost:5678>

Editor: `N` add node · **Execute step** test one node · **Execute workflow** test all · **Publish** (`Shift` + `P`) go live · `Ctrl`/`Cmd` + `K` command bar

Expressions: `{{ $json.field }}` · `{{ $('Node name').item.json.field }}` · `{{ $now.toFormat('yyyy-MM-dd') }}` · `{{ $ifEmpty($json.x, 'default') }}`

Code node: `$input.all()` (all items) · `$input.item.json` (each item) · return `[{ json: {...} }]`
· Python: `_items`, `item["json"]["field"]`

Webhooks: test URL while listening · production URL only after publishing · secure with header or basic auth

Reliability: Executions tab · Debug in editor · Retry On Fail · error output · error workflow with Error Trigger · pin data while building

AI: AI Agent = chat model + memory + tools + system message · RAG = embeddings + vector store + splitter · MCP Client Tool to use servers · MCP Server Trigger to become one · instance-level MCP for Claude and ChatGPT

Self-hosting: Compose + PostgreSQL + Caddy · `N8N_WEBHOOK_URL`, `N8N_PROXY_HOPS=1`, `N8N_ENCRYPTION_KEY` · back up `.n8n` volume and database · update: edit version, `docker compose pull`, `docker compose up -d` :::

Glossary

AI Agent node : A workflow node that uses an AI model to decide which attached tools to call to complete a request.

API : A way for programs to talk to each other; n8n's HTTP Request node can call almost any API.

Binary data : File data, such as a PDF or image, carried in an item's `binary` section.

Caddy : A reverse proxy that obtains and renews HTTPS certificates automatically.

Chat model : The sub-node that connects an AI node to a language model provider.

Chunking : Cutting long documents into smaller passages so they can be searched separately.

Cluster node : An AI root node, such as AI Agent, with sub-nodes plugged into it.

Concurrency : How many executions can run at the same time.

Credential : Saved sign-in details, an OAuth connection or an API key, that let nodes use an app.

Cron : A compact text format for schedules, such as `0 9 * * 1-5`.

Data table : A simple table stored inside n8n for keeping data between workflow runs.

DNS record : A setting that points a domain name at a server's IP address.

Docker : Software that runs programs in sealed containers.

Docker Compose : A tool that starts several containers together from one file.

Draft : Saved edits to a workflow that aren't live until you publish.

Embedding : A list of numbers representing the meaning of a piece of text.

Encryption key : The secret n8n uses to encrypt saved credentials.

Environment variable : A setting passed to n8n when it starts, such as `N8N_WEBHOOK_URL`.

Error workflow : A workflow starting with an Error Trigger that runs when another workflow fails.

Execution : One run of a workflow, logged with all its data.

Expression : A formula inside `{{ }}` that inserts or calculates values from earlier nodes.

Gateway credits : Prepaid n8n Cloud credits for running supported AI models without your own provider account.

Guardrails : A node that checks text for unsafe content, personal data or off-topic requests.

Hallucination : When an AI model confidently states something false.

HTTPS : Encrypted web traffic, shown by a padlock in the browser.

Instance : Your own copy of n8n, with its own address.

Item : A single piece of data, such as one row or one email, flowing through a workflow.

Item linking : n8n's record of which input item produced each output item.

JSON : The structured text format n8n uses to represent data.

LLM : Large language model, the AI that reads and writes text.

Luxon : The date library behind `$now` and `$today`.

MCP : Model Context Protocol, the standard for connecting AI apps to tools; see the *MCP* manual.

Memory : A sub-node that stores an AI conversation so the agent remembers it.

Node : One building block in a workflow: a trigger, an app action, or a logic step.

OAuth : A secure “sign in with...” method for connecting apps without sharing your password.

Pin data : Freeze a node's output so you can keep building without re-running it.

Polling : A trigger that checks an app on a timer for new data.

PostgreSQL : A robust database recommended for production n8n.

Production execution : A run started by a trigger on a published workflow; the kind plans count.

Publish : Make the current version of a workflow live so its triggers run automatically.

Queue mode : A scaling setup where workers run executions taken from a Redis queue.

RAG : Retrieval-augmented generation: the AI looks up your documents before answering.

Rate limit : A cap on how many requests a service accepts in a period.

Redis : A fast data store used as the queue in queue mode.

Reverse proxy : A program that receives web traffic, handles HTTPS and passes requests to n8n.

Self-hosting : Running n8n on your own computer or server instead of n8n Cloud.

Sub-workflow : A workflow called by another workflow, used to reuse common steps.

System message : Standing instructions that tell an AI who it is and how to behave.

Task runner : A separate process where n8n runs Code node code safely.

Token : A chunk of text, about three-quarters of a word, that AI providers charge by.

Trigger : The node that starts a workflow, such as a schedule, webhook, form, or app event.

Two-factor authentication : Signing in with a password plus a code from your phone.

Vector store : A database that finds text by similarity of meaning.

VPS : A virtual private server: a rented, always-on computer in a data centre.

Webhook : A unique web address that other apps send data to, which starts a workflow.

Worker : An n8n process that runs executions in queue mode.

Workflow : A set of connected nodes that automates a task.

Workflow history : Saved earlier versions of a workflow that you can view or restore. :::