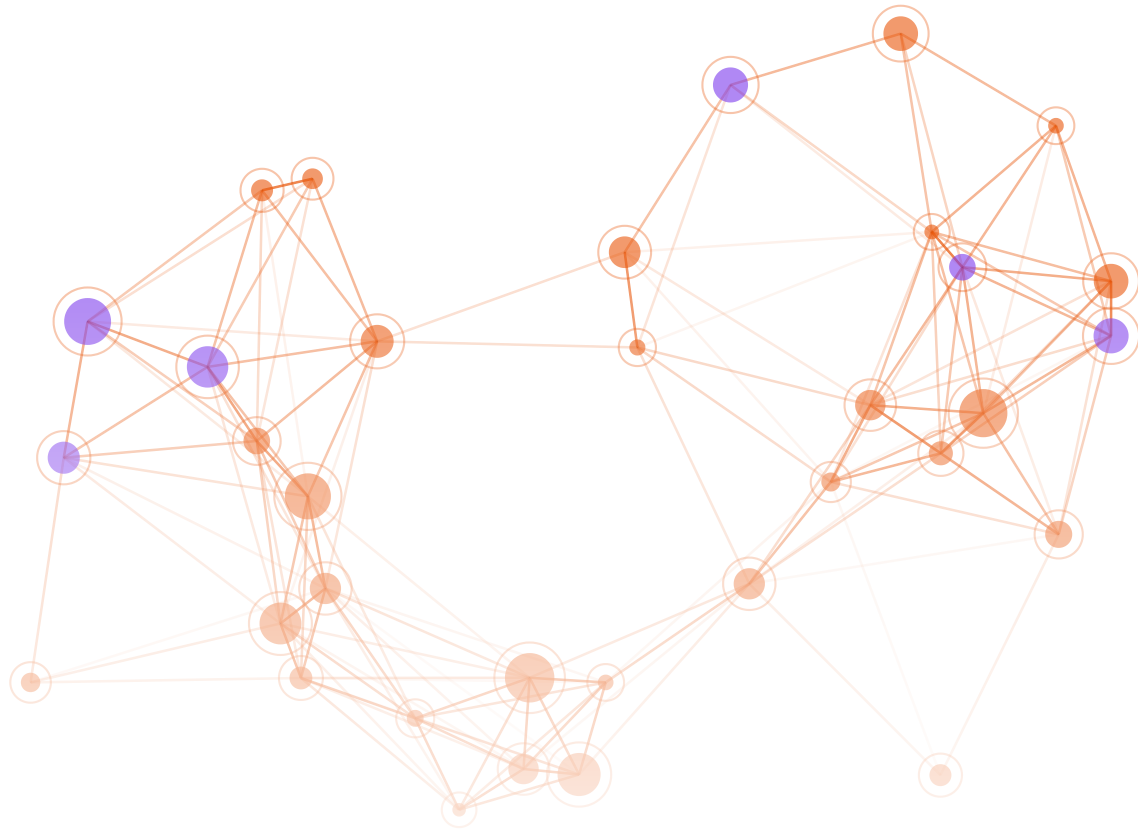




A BEGINNER'S MANUAL

GROK EDITION

Zapier & Make

Connect your apps, automate the boring parts, and add AI to everyday workflows without writing code

Zapier & Make

Connect your apps, automate the boring parts, and add AI to everyday workflows without writing code

GROK EDITION Written by Grok Bot, with facts checked against official sources in October 2026.

Zapier and **Make** are the two best-known no-code automation platforms. Both connect thousands of apps, such as Gmail, Google Sheets, Slack, Notion and your CRM, so that something happening in one app automatically does something in another: "when a form is submitted, add a row to my spreadsheet and send me a Slack message."

This manual teaches both side by side. You will learn how each one counts and charges for usage (Zapier **tasks**, Make **credits**), build your first workflow in each, master filters, branches, loops, webhooks and code steps, add AI that summarizes, classifies and drafts, connect AI chat tools through MCP, and keep everything reliable, secure and affordable.

Plan prices and limits were checked on the official Zapier and Make sites on October 16, 2026.

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.

Prices, plan limits and feature availability were checked on zapier.com, help.zapier.com, make.com and help.make.com on October 16, 2026 (US prices, before tax). Both companies change plans, prices and features regularly; confirm on the official pricing pages before buying. Menu names in this manual may differ slightly as the products update. Automations act on real data in your accounts; test with sample data first.

Contents

01	Meet Zapier and Make	5
1.1	Triggers and actions	5
1.2	How triggers find new data	6
1.3	The vocabulary, side by side	6
1.4	How they feel different	7
1.5	What you'll build in this manual	8
02	Prices and Usage, Explained	9
2.1	Zapier plans (checked October 16, 2026)	9
2.2	Make plans (checked October 16, 2026)	10
2.3	Worked cost example	12
2.4	Seven ways to use fewer tasks and credits	13
03	Your First Zap	14
3.1	Create your account	14
3.2	A tour of the Zap editor	14
3.3	The example: form to sheet to Slack	15
3.4	Reading Zap history	16
3.5	Editing a live Zap	16
04	Zapier Building Blocks	17
4.1	Filter: continue only if...	17
4.2	Paths: if this, do that; otherwise...	17
4.3	Formatter: clean up the data	18
4.4	Delay: wait before continuing	18
4.5	Looping: once per item	18
4.6	Webhooks by Zapier	18
4.7	Code by Zapier	19
4.8	Tables and Forms	20
05	AI Inside Zapier	21
5.1	AI steps in a Zap	21
5.2	Copilot	22
5.3	Agents	22
5.4	Chatbots	23
5.5	Canvas	23
5.6	Zapier MCP: give your AI hands	23
06	Your First Make Scenario	25
6.1	Create your account	25

6.2	A tour of the scenario editor	25
6.3	Bundles: the idea that makes Make click	26
6.4	The example: form to sheet to Slack, the Make way	26
6.5	Scenario history and incomplete executions	27
07	Make Building Blocks	28
7.1	Filters	28
7.2	Router and routes	28
7.3	Iterators and aggregators	29
7.4	Webhooks	29
7.5	The HTTP app	29
7.6	Data stores	29
7.7	Error handlers	30
7.8	The Code app	30
08	AI Inside Make	31
8.1	Two ways to pay for AI	31
8.2	AI modules in a scenario	32
8.3	Make AI Agents	32
8.4	Make MCP server	32
09	Designing Good Automations	34
9.1	Write it as a sentence first	34
9.2	The build loop	34
9.3	Naming and organizing	35
9.4	Avoiding duplicates	35
9.5	Prompting for automations	35
10	Ten Recipes	37
11	Reliability, Monitoring and Maintenance	40
11.1	Why automations fail	40
11.2	Monitoring routine	41
11.3	Recovering from an outage	41
12	Security and Privacy	42
12.1	Connections and permissions	42
12.2	Sensitive data	43
12.3	Prompt injection in automations	43
13	Zapier, Make or n8n?	44
14	Troubleshooting	45
	Quick Reference	46
	Glossary	48

Meet Zapier and Make

- An **automation** watches for an event in one app (the **trigger**) and then does work in other apps (the **actions**).
- **Zapier** calls an automation a **Zap**. **Make** calls it a **scenario**.
- Both are cloud services: your automations run on their servers, even when your computer is off.
- Both have free plans, so you can learn everything in this manual for \$0 before deciding whether to pay.

Think about the small, repetitive jobs you do every week. You copy a new customer's details from an email into a spreadsheet. You forward invoices to your accountant. You post the same announcement to Slack and to Teams. You check a website for changes. Each takes two minutes, but together they eat hours, and you sometimes forget one.

An automation platform does these jobs for you. You describe the job once, as a series of steps, and the platform runs it every time the starting event happens, day or night.

1.1 Triggers and actions

Every automation in both tools has the same shape.

1

Trigger.

The event that starts the automation. “A new email arrives with a label,” “a new row is added to a sheet,” “a form is submitted,” or “every Monday at 9 a.m.”

2

Actions.

The work that follows. “Create a task,” “send a message,” “add a row,” “ask an AI to summarize.”

3 Data flowing between them.

Each step produces data (an email's subject, sender and body) that later steps can use. You **map** those values into the fields of the next step.

NOTE

You never have to write code to do this. You pick apps from a menu, sign in to them once, and choose fields from drop-down lists. Code steps exist for the rare cases where you need them, and this manual shows you how to use them safely.

1.2 How triggers find new data

There are two ways a trigger learns that something happened, and the difference matters for speed and cost.

TRIGGER TYPE	HOW IT WORKS	SPEED
Polling	The platform asks the app "anything new?" on a timer	Every 1 to 15 minutes, depending on your plan
Instant (webhook)	The app tells the platform the moment something happens	Seconds
Schedule	Runs at times you set, with no outside event	When you choose

On Zapier's Free plan, polling triggers check every 15 minutes; paid plans check every 2 minutes (Pro) or every minute (Team). On Make's Free plan the minimum interval between scheduled runs is 15 minutes; paid plans allow 1 minute. Instant triggers fire immediately on any plan, so prefer them when an app offers one.

1.3 The vocabulary, side by side

The two products describe the same ideas with different words. Keep this table handy.

IDEA	ZAPIER	MAKE
An automation	Zap	Scenario
One building block	Step	Module
A single run	Zap run	Execution (run)
Usage unit you pay for	Task	Credit
Branching	Paths	Router
Stop unless a condition is true	Filter	Filter (on the line between modules)
Run history	Zap history	Scenario history
Saved login to an app	App connection	Connection
Built-in database	Tables	Data stores
Receive data from anywhere	Webhooks by Zapier	Webhooks app (custom webhook)

1.4 How they feel different

Zapier is built like a checklist. You add steps from top to bottom in a sidebar. It is the gentlest introduction to automation, and it has the largest catalog: more than 9,000 apps according to Zapier's site in October 2026.

Make is built like a whiteboard. Modules are circles on a canvas, connected by lines, and you can see data branch, loop and merge. It takes a little longer to learn, but complex logic is easier to see, and Make's usage is usually cheaper per step. Make's pricing page lists more than 3,500 standard apps.

IF YOU...	START WITH
Want the fastest path to a working automation	Zapier
Use a niche app (check both catalogs first)	Whichever lists it
Need lots of branches, loops or data reshaping	Make
Run high volumes and watch costs closely	Make (but do the math in the next chapter)
Want an AI chat tool to act in thousands of apps	Either: both offer MCP servers
Want to self-host for free	Neither; see Volume 6 on n8n

TIP

You do not have to choose forever. Many people use Zapier for quick, simple automations and Make for one or two complex ones. Both free plans can sit side by side.

1.5 What you'll build in this manual

- A form-to-spreadsheet-to-Slack workflow in Zapier
- The same workflow in Make, so you can compare
- A daily AI news digest that lands in your inbox
- An AI email triage that labels and drafts replies
- Invoice and receipt filing to cloud storage
- A lead-capture pipeline with branching
- Webhook-powered automations you can trigger from anything
- AI chat tools (Claude, ChatGPT and others) taking actions through Zapier MCP and Make's MCP server

CHAPTER 2

Prices and Usage, Explained

- **Zapier charges by tasks.** A task is one successful action step. Triggers, built-in tools like Filter and Formatter, and failed steps do not count.
- **Make charges by credits.** Most module actions cost 1 credit. Routers and error handlers are free; AI and code features can cost more.
- Annual billing is cheaper on both. Check the official pricing pages before you buy; the figures below were read on October 16, 2026.

Understanding the usage unit is the single most important skill for keeping automation affordable. A badly designed workflow can cost ten times what a well-designed one does for the same result.

2.1 Zapier plans (checked October 16, 2026)

PLAN	PRICE (US)	TASKS PER MONTH	KEY LIMITS
FREE Free	\$0	100	Two-step Zaps only (one trigger, one action); 15-minute polling
Professional	From \$19.99/mo billed annually, or \$29.99 month to month	750	Multi-step Zaps, 2-minute polling, premium apps, webhooks, AI by Zapier, filters and paths
Professional, larger tiers	\$39 annually / \$58.50 monthly (1,500 tasks); \$49 / \$73.50 (2,000 tasks)	1,500 to 2,000 and up	Same features, more tasks
Team	From \$69/mo annually or \$103.50 monthly	2,000	Up to 25 users, 1-minute polling, SAML single sign-on, shared workspaces
Enterprise	Custom quote	Custom	Advanced admin and security

More facts from Zapier's pricing page:

- Annual billing saves 33% versus monthly.
- Professional comes with a 14-day trial; nonprofits can get 15% off.
- Your task allowance resets every month.
- If you go over, **pay-per-task** billing charges extra tasks at 1.25× your plan's base per-task rate on annual plans, or 2.5× on monthly plans. Usage pauses once overage reaches three times your subscription price.
- Code by Zapier steps can run for 1 second on Free and 30 seconds on paid plans by default. Extended runtimes (up to 10 minutes depending on plan) are being phased in and billed separately from June 15, 2026.
- Zapier Tables: 2,500 records on Free, 100,000 on Professional.
- Agents and Chatbots have their own allowances: Agents use **activities** (400 per month on the free Agents plan and 1,500 on the paid Agents plan), and Free accounts include 2 chatbots.

What counts as a Zapier task

COUNTS AS A TASK	DOES NOT COUNT
A successful action step, such as "Create row" or "Send message"	The trigger step
Each action run inside a loop (once per item)	Built-in tools: Filter, Paths, Formatter, Delay, Looping, Tables, Forms and similar
Each Zapier MCP tool call: 2 tasks	Steps that fail with an error

A Zap with a trigger, a Filter, a Formatter step and two app actions uses **2 tasks** per run. If the Filter stops the run, it uses **0**.

2.2 Make plans (checked October 16, 2026)

Make bills in **credits**. Since August 27, 2026, credits have been Make's billing unit. Prices scale with how many credits you buy; the table shows the entry tier of 10,000 credits per month. Make's pricing page loads these prices dynamically, so we read them from the page's own US price data on October 16, 2026; treat them as a guide and confirm on make.com before buying.

PLAN	10,000 CREDITS/MO, BILLED MONTHLY	BILLED ANNUALLY (PER MONTH)	NOTABLE FEATURES
FREE Free	\$0 (1,000 credits)	n/a	2 active scenarios, 15-minute minimum interval
Core	\$16	\$12	Unlimited active scenarios, 1-minute interval, Make API at 60 calls/min
Pro	\$28	\$21	More advanced features; Make API at 120 calls/min
Teams	\$51	\$38	Built for teams; Make API at 240 calls/min
Enterprise	Custom	Custom	Advanced security, enterprise apps, on-premises agent

WARNING

Many third-party articles still quote older, lower Make prices. Those numbers are out of date. Always use make.com/en/pricing, and be aware that larger credit tiers cost more: for example, Core at 20,000 credits was \$28 per month (monthly billing) on our check date.

More facts from Make's pricing page and help center:

- **Credits expire at the end of the term.** On monthly billing you get a monthly allocation. On annual Pro and Teams plans you get the whole year's credits up front to use flexibly; Core always allocates monthly, even on annual billing.
- Run out early? Buy **extra credits** in bundles of 1,000 or 10,000, or turn on auto-purchase (Core, Pro and Teams). Extra credits expire like regular ones.
- When you run out, scenarios stop. Incoming webhooks are queued (up to your queue limit) and processed when credits are added.
- For every 10,000 credits you get 5 GB of data transfer, 10 MB of data storage and a webhook queue of 667 items (10,000 maximum).
- The **Start-up Plan** offers 50% off Business for qualifying companies with fewer than 20 employees. Nonprofits can contact Make for discounts.

What costs Make credits

USES CREDITS	FREE
Each time a module reads, searches, creates, updates or deletes data	Router module
Transforming data with built-in tools, aggregating and iterating	Error handlers: Rollback, Break, Resume, Commit, Ignore
Make's Code app: 2 credits per second of execution	
AI features that use Make's AI Provider: extra credits based on AI tokens	

A Make scenario counts credits for **every module, every time it processes a bundle**. A bundle is one item of data. If a trigger returns 10 new emails, every following module runs 10 times. This is the most common surprise on a Make bill.

2.3 Worked cost example

You want new web-form leads saved to a Google Sheet, posted to Slack and emailed to you. That is one trigger and three actions, and you expect 300 leads a month.

	ZAPIER	MAKE
Usage per run	3 tasks (the trigger is free)	About 4 credits (the trigger module counts too)
Monthly usage	900 tasks	About 1,200 credits
Fits on	Professional 1,500-task tier (\$39/mo annual)	Core 10,000-credit tier (\$12/mo annual)

The difference grows with volume and with workflow complexity. But price is not everything: if an app you need works better on one platform, or your time to build matters more, that can outweigh the subscription.

TIP

Before you pay for either tool, build your workflow on the free plan and run it for a week. Then look at real usage in Zapier's Zap history or Make's scenario history and multiply.

2.4 Seven ways to use fewer tasks and credits

- ☐ Put a **Filter** as early as possible so runs you do not need stop before costly steps.
- ☐ Use **instant triggers** instead of frequent polling when you have the choice.
- ☐ Combine several small edits into one “update record” step instead of several.
- ☐ In Make, check how many bundles a trigger returns and set a sensible maximum.
- ☐ Use **digest** or batching patterns: collect items, then send one summary instead of one message per item.
- ☐ Turn off automations you no longer need; a forgotten Zap still counts tasks.
- ☐ Review your top task or credit users monthly and redesign the expensive ones.

Your First Zap

- Sign up at zapier.com, then build a Zap: **trigger, test, action, test, publish**.
- Always test each step with real sample data before turning a Zap on.
- After publishing, watch **Zap history** to see every run and its status.

3.1 Create your account

- 1 Go to **zapier.com** and choose **Sign up**. You can sign up with Google, Microsoft or an email address.
- 2 Answer the short onboarding questions (they only tailor suggestions).
- 3 You land on the Zapier home page, where you can create Zaps, Tables, Forms, Agents and more.

TIP

Use the same work or personal account you will connect apps from. It keeps app connections and billing in one place.

3.2 A tour of the Zap editor

The Zap editor has three areas.

- **The canvas** in the middle shows your steps from top to bottom: the trigger first, then each action.
- **The step panel** on the right opens when you click a step. It has tabs for choosing the app and event, picking an account, configuring fields and testing.
- **The top bar** holds the Zap's name, its on/off toggle and the **Publish** button.

3.3 The example: form to sheet to Slack

You run a small business. When someone fills in your contact form, you want their details in a Google Sheet and a notification in Slack. Free plans only allow two-step Zaps, so on Free you can build the sheet part; the Slack step needs Professional or the trial.

Step 1: the trigger

- 1 Click **Create** and choose **Zap**.
- 2 Click the **Trigger** step and search for your form app (Google Forms, Typeform, Jotform or Zapier's own Forms).
- 3 Choose the **event**, for example "New Form Response."
- 4 Choose or connect the **account**. A sign-in window opens; approve access.
- 5 Pick which form to watch.
- 6 Click **Test trigger**. Zapier fetches a recent response so later steps have sample data. Submit a test entry first if the form has none.

Step 2: the first action

- 1 Click + below the trigger and choose **Google Sheets**.
- 2 Choose the event **Create Spreadsheet Row**.
- 3 Connect your Google account, then pick the spreadsheet and worksheet.
- 4 Zapier shows one field for each column header. Click into a field and choose a value from the trigger, such as **Name** or **Email**. This is called **mapping**.
- 5 Click **Test step**. Check the sheet: a new row should appear.

☆ IMPORTANT

Test steps do real work. Testing a "Send email" step sends a real email; testing "Create row" creates a real row. Use your own address and test data, and delete test records afterward.

Step 3: the Slack action

- 1 Add another step, choose **Slack**, and pick **Send Channel Message**.

- 2 Choose the channel, then build the message text by mixing typed words with mapped fields: “New lead: [Name] ([Email]) wrote: [Message]”.
- 3 Test the step and check Slack.

Step 4: name and publish

Give the Zap a clear name, such as “Contact form → Sheet + Slack,” then click **Publish**. The Zap is now on. Zapier will check the form every 15 minutes on Free (2 on Professional) or receive it instantly if the trigger is instant.

3.4 Reading Zap history

Every time a Zap runs, Zapier records it in **Zap history**. Open it from the left sidebar.

STATUS	WHAT IT MEANS	WHAT TO DO
Success	Every step worked	Nothing
Filtered	A Filter stopped the run on purpose	Nothing, unless you expected it to continue
Errored	A step failed	Open the run, read the error, fix the step, then replay
Halted	A step stopped the run without a full error	Read the details; often missing data
Held / On hold	Zapier paused the run, for example for review	Check the reason; replay on a paid plan

You can filter history by date, Zap name, app, folder, owner and status, and export up to 5,000 runs at a time to CSV or JSON.

Replay and Autoreplay

When a step errors because an app was down, you do not need to resubmit anything. Select the errored runs in Zap history and **replay** them. On the Free plan, manual replay works for errored runs; replaying other statuses, such as on-hold runs, needs a paid plan. You can also turn on **Autoreplay**, which retries errored runs automatically.

3.5 Editing a live Zap

When you edit a published Zap, Zapier creates a draft. The live version keeps running until you publish the changes. This lets you safely tweak a working automation.

TIP

Keep a short description in each Zap: what it does, who asked for it and what it depends on. Six months from now you will not remember why it exists.

Zapier Building Blocks

- **Filter** stops runs that don't match. **Paths** sends runs down different branches.
- **Formatter** cleans text, dates and numbers. **Delay** waits. **Looping** repeats steps for each item.
- **Webhooks by Zapier** sends and receives data from anything with a URL. **Code by Zapier** runs short Python or JavaScript.
- **Tables** stores data; **Forms** collects it. None of these built-in tools count as tasks.

Multi-step Zaps and these tools need a paid plan (or the trial), except where noted.

4.1 Filter: continue only if...

A Filter step checks a condition and stops the run if it is false. For example: continue only if the email subject contains “Invoice,” or only if the deal value is greater than 1,000.

You can combine rules with **And** (all rules must be true) or **Or** (at least one must be true).

Example: “Subject contains *Urgent* **and** *help*” continues only when both words appear.

4.2 Paths: if this, do that; otherwise...

Paths splits a Zap into branches, each with its own rules and steps. Use Paths when you want different outcomes, and a Filter when you only want to stop.

USE	WHEN
Filter	“Only handle invoices; ignore everything else”
Paths	“Invoices go to Accounting; support emails go to the help desk; everything else gets a label”

TIP

Give each path a descriptive name like “Path A: Invoice.” When you look at Zap history later, the names tell you which way each run went.

4.3 Formatter: clean up the data

Formatter by Zapier transforms values between steps without code. Common jobs:

- **Text:** capitalize a name, trim spaces, extract an email address or URL from a block of text, split “First Last” into two parts, find and replace.
- **Numbers:** format currency, do simple math, convert a “1,234.50” string to a number.
- **Dates:** convert “10/16/2026” into “2026-10-16,” add three days, change time zones.
- **Utilities:** look up a value in a small table, pick an item from a list, convert line items.

4.4 Delay: wait before continuing

Delay can pause a Zap for a set time (“wait 1 day”), until a time (“wait until 9 a.m. tomorrow”), or queue runs so they happen one after another. A classic use is a follow-up email: on sign-up send a welcome, then wait three days and send tips.

4.5 Looping: once per item

If a trigger gives you a list, such as the line items of an order, **Looping by Zapier** runs the following steps once per item. Remember: each action inside the loop counts as a task for each item.

4.6 Webhooks by Zapier

A **webhook** is a URL that receives data. Webhooks by Zapier can:

- **Catch Hook:** give you a unique URL. Anything that sends data to it (a website, a script, another tool) starts your Zap instantly.
- **POST, GET, PUT:** send data to any web address, useful for apps Zapier doesn’t list.

Data usually arrives as JSON (`{"name": "Alex", "age": 27}`) or form-encoded text (`name=Alex&age=27`). Zapier parses both into fields you can map.

Rate limits apply: Zapier’s help center lists 20,000 requests every 5 minutes per user, and returns error code 429 above that. Professional, Team and Enterprise users can send requests from a static IP address if the receiving server requires an allowlist.

⚠ WARNING

Treat a Catch Hook URL like a password. Anyone who has it can trigger your Zap. Don’t post it publicly, and add a Filter that checks for a secret value you include in every legitimate request.

4.7 Code by Zapier

When no built-in tool does what you need, a Code step runs a small script. Zapier's help center lists **Python 3.13** and **JavaScript on Node.js 22**. You pass in values through the **Input Data** fields (you cannot map fields directly into the code), and the step returns an `output` dictionary whose keys become fields for later steps.

CODE BY ZAPIER (RUN PYTHON): CLEAN AND SCORE A LEAD

```
# Input Data fields set in the step: name, email, message
name = input_data.get("name", "").strip().title()
email = input_data.get("email", "").strip().lower()
message = input_data.get("message", "")

hot_words = ["quote", "pricing", "buy", "urgent", "demo"]
score = sum(1 for w in hot_words if w in message.lower())

output = {
    "name": name,
    "email": email,
    "domain": email.split("@")[-1] if "@" in email else "",
    "score": score,
    "is_hot": score >= 2,
}
```

Facts to remember about Code steps:

- They include the Python standard library, `requests` and BeautifulSoup. You cannot install other packages.
- Dates and times are in **UTC**, regardless of your account's time zone.
- Default runtime is 1 second on Free and 30 seconds on paid plans.
- Returning a **list** of dictionaries makes the following steps run once per item.
- Zapier recommends building a custom integration instead if you need more than two HTTP requests or authenticated requests.

TIP

You do not need to know Python to use Code steps. Describe what you want to an AI assistant (“Write a Code by Zapier Python step. Input fields: name, email. Output: name in title case, email lowercase”) and paste the result. Then test it with sample data. Volume 14 teaches the basics of Python.

4.8 Tables and Forms

Zapier Tables is a built-in database: rows and columns that Zaps can read, write and update. Use it to remember things between runs, like which leads you have already contacted. Free accounts can store 2,500 records; Professional, 100,000. Tables can also **enrich** records with AI: one prompt fills several columns, such as a summary and a category.

Zapier Forms (renamed from Interfaces) builds forms that start Zaps: lead capture, intake requests, approvals, surveys, sign-ups. Forms is free, and Forms triggers and actions do not count toward your task usage. Existing Interfaces pages, layouts and data connections keep working.

AI Inside Zapier

- **AI by Zapier** adds an AI step to any Zap to summarize, classify, extract or draft.
- **Copilot** builds and edits Zaps from plain-language requests (open beta).
- **Agents** are AI teammates that act in your apps; **Chatbots** answer questions from your content.
- **Zapier MCP** lets Claude, ChatGPT, Cursor and other AI tools take actions in 9,000+ apps; each tool call costs 2 tasks.

5.1 AI steps in a Zap

The AI by Zapier step (on paid plans) sends text to an AI model with your instructions and returns the result for later steps. You can also use the ChatGPT (OpenAI), Anthropic (Claude) or Google AI apps with your own API key, which bills your AI provider account separately; see Volume 15 for API costs.

Four jobs AI steps do well:

JOB	EXAMPLE INSTRUCTION
Summarize	"Summarize this email in two sentences for a busy manager."
Classify	"Label this message as one of: Sales, Support, Billing, Spam. Reply with the label only."
Extract	"Return the invoice number, total and due date as JSON."
Draft	"Write a friendly two-paragraph reply. Do not promise refunds or dates."

PROMPT

You are a triage assistant for a small design studio.

Read the email below and reply with JSON only, no other text:

```
{"category": "Sales" or "Support" or "Billing" or "Other",  
  "urgency": "high" or "normal",  
  "summary": "one sentence",  
  "needs_human": true or false}
```

Rules:

- "high" urgency only for outages, deadlines within 48 hours, or angry customers.
- needs_human is true for anything about money, contracts or complaints.

Email:

[mapped: Body Plain]

☆ IMPORTANT

Ask for **one exact format**, such as a single label or JSON with fixed keys. The following Filter or Paths step can then reliably check the value. Free-form AI answers break branches.

5.2 Copilot

Zapier Copilot is an AI assistant that works across Zapier's products. Describe what you want ("When a customer pays in QuickBooks, find their open deal in HubSpot and mark it closed") and Copilot outlines or builds the Zap step by step, picks app connections, and can add or replace steps in Zaps you already have. Zapier labels Copilot an open beta, so review every step it creates before publishing.

5.3 Agents

Zapier Agents are AI assistants that use your connected apps to do work: research a lead and update the CRM, or prepare a weekly report. You give an agent instructions, a trigger, and the apps and actions it may use. For each action field you choose whether the agent always uses a fixed value, picks from a set of values, guesses a value, or leaves it blank.

Agents are billed in **activities** (any billable action an agent takes). Anyone can try Agents free; Team accounts share one pool of activities. Zapier notes that Agents doesn't yet support the app and action restrictions some Enterprise accounts use.

💡 TIP

Start agents in the narrowest possible role: read-only access, one app, one job. Give them write access only after you have watched a week of runs.

5.4 Chatbots

Zapier Chatbots answer questions using knowledge sources you provide (documents, websites, Tables) and can trigger actions, such as capturing a lead into a Zap. Free accounts include two chatbots.

5.5 Canvas

Zapier Canvas is a diagramming tool for planning workflows. Sketch the whole process, including the human steps, and Canvas helps you turn parts of it into Zaps.

5.6 Zapier MCP: give your AI hands

The **Model Context Protocol (MCP)** is an open standard that lets AI apps use outside tools (see Volume 5). **Zapier MCP** gives AI tools such as Claude, ChatGPT and Cursor access to more than 30,000 actions across 9,000+ apps through one connection.

ZAPS	ZAPIER MCP
Run automatically when a trigger fires	Run when you ask your AI to do something
Fixed steps you designed	The AI decides which allowed action to call
Uses tasks per action	Uses 2 tasks per tool call

Set it up

- 1 Go to **zapier.com/mcp** and start the guided setup. Zapier says it takes about five minutes, with no code or config files, and recommends Claude as the easiest starting point.
- 2 Choose which apps and which actions the AI may use. Prefer read-only actions to begin with.
- 3 Connect the MCP server to your AI tool by following the instructions for that tool.
- 4 In your AI chat, ask for something that needs the action: “Find my next three Google Calendar events.”
- 5 Check the **History** tab in Zapier: every MCP action is logged there.

 DANGER

An AI with MCP access can send emails and change records in your name. Allow only the actions you need, avoid “send” and “delete” actions until you trust the setup, and remove access from any app you stop using. Be especially careful when the AI is also reading untrusted content, such as incoming emails or web pages, which might contain hidden instructions (prompt injection, covered in Volume 18).

Your First Make Scenario

- Sign up at make.com and create a **scenario**: modules connected left to right on a canvas.
- **Run once** to test with live data, then set a **schedule** and switch the scenario on.
- Each module shows a bubble with the number of operations it ran; click it to inspect every **bundle** of data.

6.1 Create your account

- 1 Go to **make.com** and choose **Get started free**. Sign up with Google, Microsoft or email.
- 2 Make creates an **organization** (your billing account) and a **team** inside it. Scenarios live in teams.
- 3 From the left menu, open **Scenarios** and click **Create a new scenario**.

6.2 A tour of the scenario editor

- **The canvas** is where modules sit. Each module is a circle with the app's icon. Lines between them show the order data flows.
- **The big plus** in the middle adds your first module, which is always the trigger.
- **The toolbar at the bottom** has **Run once**, the scheduling switch, scenario settings, and tools such as Flow control (routers, iterators, aggregators) and other built-in tools.
- **Bubbles** appear above modules after a run, showing how many operations each did. Click one to see the exact input and output data.

6.3 Bundles: the idea that makes Make click

Make passes data between modules as **bundles**. A bundle is one item: one email, one row, one order. If a trigger finds 5 new rows, it outputs 5 bundles, and every module after it runs once per bundle. That is powerful (no loop setup needed) and it is also why credit usage multiplies.

☆ IMPORTANT

Most triggers have a **Limit** field: the maximum number of bundles to fetch per run. Set it on purpose. A limit of 2 is perfect while testing.

6.4 The example: form to sheet to Slack, the Make way

This is the same workflow you built in Zapier, so you can compare.

Step 1: the trigger module

- 1 Click the big plus and search for your form app, for example **Google Forms** or **Typeform**. Choose its “watch responses” trigger.
- 2 Click **Create a connection** and sign in. Make saves the connection for reuse.
- 3 Choose the form. When asked where to start, choose to start from now on, or pick a recent response for testing.
- 4 Set **Limit** to 2 for testing. Click **OK**.

Step 2: add Google Sheets

- 1 Hover over the right edge of the trigger and click the small plus to add a module. Choose **Google Sheets > Add a Row**.
- 2 Connect Google, choose the spreadsheet and sheet, and say whether the sheet has headers.
- 3 In each column field, click to open the **mapping panel** and choose values from the trigger: Name, Email, Message.
- 4 Click **OK**.

Step 3: add Slack

Add **Slack > Create a Message**, pick the channel, and compose the text with mapped values, exactly as in Zapier. Slack works on Make's Free plan because Make does not limit the number of steps.

Step 4: test with Run once

Click **Run once** and submit a test form entry (or let it fetch the existing one). Watch the bubbles appear. Click the bubble over each module to see what went in and what came out. If something is wrong, fix the module and run once again.

Step 5: schedule and activate

- 1 Click the clock on the trigger module, or the scheduling setting in the toolbar.
- 2 Choose how often to run: at regular intervals (every 15 minutes on Free, down to every minute on paid plans), once a day, on specific days, or **immediately** for instant triggers.
- 3 Save the scenario (Ctrl+S or the save icon), give it a name, then switch it **on**.

NOTE

The Free plan allows **two active scenarios** at once. You can build more and switch them on and off as needed, or upgrade to Core for unlimited active scenarios.

6.5 Scenario history and incomplete executions

Open a scenario and choose **History** to see every run: when it started, how long it took, how many credits it used, and whether it succeeded. Click a run to replay its bubbles and inspect the data.

If a module fails, Make can store the run as an **incomplete execution** so you can fix the problem and resume it rather than losing the data. Storage for incomplete executions grows with your credit tier (10 MB per 10,000 credits, up to 2 GB). Turn this on in the scenario settings.

Make Building Blocks

- **Filters** sit on the lines between modules. **Routers** split the flow into routes.
- **Iterators** split a list into bundles; **aggregators** combine bundles back into one.
- **Webhooks** start scenarios instantly; the **HTTP** app calls any web API.
- **Data stores** remember data between runs. **Error handlers** decide what happens when something fails.

7.1 Filters

In Make, a filter lives on the **link** between two modules. Click the line (or the wrench that appears) and choose **Set up a filter**. Give it a label and a condition such as “Email: Contains: @gmail.com.” Bundles that fail the condition stop at the filter. Filters can use AND and OR conditions and many operators: text, numbers, dates, existence.

7.2 Router and routes

A **router** branches the flow. Each route can have its own filter and modules. Make’s help center notes three things beginners should know:

- Routes are processed **one after another**, not in parallel. The second route runs after the first finishes.
- You can **order routes** to decide which runs first.
- A **fallback route** catches bundles that don’t match any other route’s filter.

The router module itself costs no credits.

PATTERN	ROUTER SETUP
"Do A and B for every item"	Two routes, no filters
"If invoice, do A; otherwise do B"	Route 1 filter: subject contains "invoice"; Route 2: fallback
"Route by category from an AI step"	One route per category, each filtered on the AI's label, plus a fallback

7.3 Iterators and aggregators

An **iterator** takes an array (a list) and outputs each item as its own bundle. Example from Make's help center: an email has several attachments; an iterator splits them so each one can be uploaded to Google Drive separately. Many apps include ready-made iterators, such as **Email > Iterate attachments**.

An **aggregator** does the reverse: it collects many bundles into one. Use a **text aggregator** to join 20 headlines into one digest, or an **array aggregator** to build a list you can send in a single API call. Aggregating is a great way to save credits: one email with 20 items costs much less than 20 emails.

7.4 Webhooks

Make's **Webhooks** app creates a custom URL that starts your scenario as soon as data arrives. App modules marked **INSTANT** use webhooks behind the scenes. By default the scenario runs immediately when a request arrives, but you can schedule the scenario to process waiting webhook requests in batches instead.

If you run out of credits, incoming webhooks wait in a **queue** (667 items per 10,000 credits, up to 10,000) and are processed once credits are added.

7.5 The HTTP app

When Make has no ready-made module for a service, the **HTTP** app's request module can call any web API: choose the URL, method (GET, POST and so on), headers and body. This is how you reach brand-new AI APIs or internal tools. Volume 15 explains API keys, request formats and costs.

7.6 Data stores

A **data store** is a small database inside Make. Use it to remember which items you have already processed, keep counters, or look up settings. You get 1 MB of data storage per 1,000 credits; the Free plan's 1,000 credits equal 1 MB, enough for one data store. An organization can have up to 1,000 data stores.

7.7 Error handlers

Right-click a module and choose **Add error handler** to decide what happens when it fails. Error handlers cost no credits.

HANDLER	WHAT IT DOES	USE IT WHEN
Ignore	Skips the error and ends this bundle's route quietly	Failures are harmless, for example a duplicate
Resume	Substitutes a value you set and continues	You have a sensible default
Break	Stores the run as an incomplete execution for retry	The failure is temporary, such as an app outage
Commit	Stops and keeps the changes made so far	Partial success is acceptable
Rollback	Stops and tries to undo changes in modules that support it	All-or-nothing jobs

7.8 The Code app

Make's **Code** app runs JavaScript or Python inside a scenario on paid plans. It costs **2 credits per second** of execution, so keep scripts short. For most data reshaping, built-in functions in the mapping panel (text, math, date and array functions) are cheaper and easier.

AI Inside Make

- Add AI with app modules (OpenAI, Anthropic Claude, Google Gemini and others) using your own API key, or with **Make's AI Provider**, billed in credits.
- **Make AI Agents** reason over goals and use your scenarios, modules and MCP tools as tools.
- **Make MCP server** lets Claude, ChatGPT and other AI clients run your scenarios on demand.

8.1 Two ways to pay for AI

	MAKE'S AI PROVIDER	YOUR OWN AI PROVIDER CONNECTION
Who can use it	All plans	Paid plans
How it bills	Make credits, including credits based on AI tokens	1 credit per operation in Make, plus your provider's API bill
Setup	Nothing extra	An API key from OpenAI, Anthropic or another provider
Best for	Getting started quickly	Choosing exact models and keeping AI costs on one bill

8.2 AI modules in a scenario

The simplest AI scenario has three modules: a trigger, an AI module that summarizes or classifies, and an action that uses the answer. Use the same prompt discipline as in Zapier: one job per AI step, exact output formats, and rules about what the AI must never do.

PROMPT

Classify this support message. Reply with exactly one word from this list:
billing, bug, how-to, feedback, spam
Message:
[mapped: Text]

Then put a router after the AI module with one route per word, each filtered on the AI's output, and a fallback route for anything unexpected.

8.3 Make AI Agents

Make's AI agents receive instructions, optional **knowledge** (files such as PDFs, DOCX, CSV, JSON or TXT), and **tools**. A tool can be a single module, a whole scenario, or an MCP tool. The agent decides which tools to call to reach its goal. Make's help center documents credit usage: on Make's AI Provider, running an agent costs 1 credit per operation plus credits based on AI tokens; with your own provider connection it costs 1 credit per operation, and your provider bills the tokens.

TIP

Wrap risky actions in a scenario that has its own filters and limits, then give the agent that scenario as a tool, instead of giving it raw "delete" or "send" modules. The scenario becomes a guard rail.

8.4 Make MCP server

The Make MCP server lets MCP clients such as Claude and ChatGPT run your scenarios and, with the right permissions, manage your Make account. Make's help center explains the permission **scopes**:

- **Scenario run** scopes (all plans) let the client view and run scenarios that are active and set to **on-demand** scheduling.
- **Management** scopes (paid plans) let the client view and change scenarios, teams, connections, webhooks and data stores.

You pick scopes when you connect, through an OAuth consent screen or an MCP token. Only OAuth lets you restrict access to one organization. If you want to expose only certain scenarios, use **MCP toolboxes**. Connection details are on the Make Developer Hub.

- 1 Build a scenario that does one safe job, such as “Look up an order in my sheet and return its status.” Use scenario inputs for the order number and outputs for the answer.
- 2 Set its schedule to **On demand** and activate it.
- 3 Connect your AI client to Make’s MCP server with only the scenario run scope.
- 4 Ask your AI: “What’s the status of order 1042?” It calls the scenario and answers.

 DANGER

Avoid granting management scopes to an AI client unless you truly need it to edit scenarios. With those scopes, a mistaken or manipulated AI could change or break your automations.

Designing Good Automations

- Write the process down in plain words before you build anything.
- Build in small pieces and test each one with real sample data.
- Plan for failure: missing fields, duplicates, app outages and odd AI answers.

9.1 Write it as a sentence first

Every good automation starts as one clear sentence:

When [trigger] **in** [app], **if** [condition], **then** [action] **in** [app], **and** [action] **in** [app].

“When a new email with the label *Receipts* arrives in Gmail, if it has an attachment, then save the attachment to the *Receipts 2026* folder in Google Drive, and add a row with the date, sender and file link to my *Expenses* sheet.”

If you cannot write the sentence, you are not ready to build. Ask yourself what should happen for unusual cases: no attachment, two attachments, an attachment that is not a PDF.

9.2 The build loop

- 1 **Sketch** the steps on paper or in Zapier Canvas.
- 2 **Build the trigger** and test it with a real example.
- 3 **Add one step**, test it, check the result in the target app.
- 4 **Repeat** until all steps work.
- 5 **Test the odd cases:**
empty fields, long text, special characters, an item that should be filtered out.

6 Turn it on and watch the history for the first few days.

9.3 Naming and organizing

- Name automations as **Source → Destination: purpose**, for example “Gmail → Drive: save receipts.”
- Use folders (Zapier) or scenario folders (Make) by area: Sales, Finance, Personal.
- Add a description or note listing the owner, the date built and what depends on it.
- Rename steps and modules to say what they do (“Find customer by email”) instead of the default event name.

9.4 Avoiding duplicates

Duplicates are the most common annoyance in automation. They happen when a trigger fires twice, a run is replayed, or two automations do the same job.

- ☐ Before creating a record, add a **search** step (“Find row by email”) and create only if nothing is found. Both tools offer “find or create” style actions in many apps.
- ☐ Store a unique ID (order number, email message ID) in a Zapier Table or Make data store and check it first.
- ☐ When replaying failed runs, check whether earlier steps already succeeded.
- ☐ Keep one automation per job; search your account before building a new one.

9.5 Prompting for automations

AI inside an automation runs without you watching, so prompts must be stricter than in chat.

DO	DON'T
Ask for one job per AI step	Ask for a summary, a label and a reply in one step
Fix the output format: a single word, or JSON with set keys	Accept free-form prose that later steps must interpret
List the allowed values	Hope the AI picks consistent labels
State what the AI must never do (“never promise refunds”)	Let drafted replies go out without review
Include a fallback (“if unsure, reply Other”)	Assume every input is well formed
Keep inputs short: map only the fields needed	Send whole email threads with signatures and history

 TIP

For anything that goes out in your name, have the AI write a **draft** (a Gmail draft, a Slack message to yourself, a row in a review sheet) and approve it by hand. Move to fully automatic sending only after weeks of good drafts.

Ten Recipes

Each recipe lists the steps in both tools. “AI step” means AI by Zapier or an AI app in Zapier, and an AI module or Make’s AI Provider in Make. Recipes with more than two steps need a paid Zapier plan; most fit Make’s Free plan if you stay within two active scenarios.

Recipe 1: Save email attachments to cloud storage

- **Trigger:** Gmail, new email matching search `label:receipts has:attachment`.
- **Zapier:** Gmail trigger (with attachments) → Google Drive *Upload File* with the attachment mapped → Google Sheets *Create Row* with date, sender, file link.
- **Make:** Gmail *Watch emails* → **Email > Iterate attachments** (or an iterator on the attachments array) → Google Drive *Upload a file* → Google Sheets *Add a row*.
- **Watch out:** emails with several attachments. Make’s iterator handles each one; in Zapier, check how the trigger presents multiple files and add Looping if needed.

Recipe 2: A daily AI news digest

- **Trigger:** schedule, every weekday at 7:00 a.m.
- **Steps:** RSS feed (*new items*, or *fetch recent items*) → AI step: “Summarize each headline in one line, group by topic, maximum 10 items” → Email to yourself.
- **Make advantage:** a **text aggregator** joins all items into one block before the AI step, so you pay for one AI call, not one per article.
- **Zapier option:** use **Digest by Zapier** to collect items through the day and release them once each morning.

Recipe 3: AI email triage with drafts

- **Trigger:** new email in your inbox (exclude newsletters with a search such as `category:promotions`).
- **Steps:** AI step returning JSON with category, urgency and needs_human → branch on category (Zapier Paths or Make router) → for Support: create a Gmail draft reply; for Billing: forward to accounting; for high urgency: Slack message to you.
- **Safety:** drafts only. Never auto-send AI replies to customers until you have reviewed many examples.

Recipe 4: Lead capture with scoring

- **Trigger:** website form or Zapier Forms submission.
- **Steps:** Code or Formatter step to clean name and email → AI step scoring the message 1–5 for buying intent → CRM *create or update contact* → if score ≥ 4 , Slack the sales channel; otherwise add to a nurture list.

Recipe 5: Meeting notes to tasks

- **Trigger:** a new document in a “Meeting notes” folder, or a new page in a Notion database (see Volume 7).
- **Steps:** AI step: “Extract action items as JSON: owner, task, due date (or null)” → loop over items (Zapier Looping; Make iterator after a JSON parse module) → create a task in your to-do app for each.

Recipe 6: Social post repurposing

- **Trigger:** new blog post in your RSS feed or CMS.
- **Steps:** AI step drafting a LinkedIn post and a short post for X, each in your tone → add both to a “Drafts” sheet or Notion page for review → optionally schedule with your social tool after approval.

Recipe 7: Invoice data extraction

- **Trigger:** new PDF in a “Invoices” Drive folder.
- **Steps:** extract text (many AI apps accept files; otherwise use a PDF parsing app) → AI step returning vendor, invoice number, date, total and currency as JSON → add a row to your bookkeeping sheet → if total is above a threshold, notify you.
- **Check:** compare AI-extracted totals against the document for the first few weeks. Volume 17 covers AI for personal finance and admin.

Recipe 8: Website change monitor

- **Trigger:** schedule, every morning.
- **Steps:** HTTP GET the page (Webhooks by Zapier or Make's HTTP app) → compare with yesterday's copy stored in Tables or a data store → if different, AI step: "Describe what changed in two bullets" → email yourself → save today's copy.

Recipe 9: Webhook from a script or shortcut

- **Trigger:** Catch Hook (Zapier) or custom webhook (Make).
- **Use:** call the URL from a phone shortcut, a spreadsheet button, or a Python script to start any workflow you like.

TRIGGER A WEBHOOK FROM PYTHON

```
import requests

WEBHOOK_URL = "https://hooks.example.com/your-unique-url" # paste yours
payload = {"secret": "change-me", "task": "Buy printer paper", "priority": "normal"}

r = requests.post(WEBHOOK_URL, json=payload, timeout=15)
print(r.status_code, r.text)
```

- **Safety:** keep the URL private and check the `secret` field with a filter as the first step.

Recipe 10: Ask your AI chat to run a workflow

- **Zapier MCP:** connect Claude or ChatGPT, allow "Google Sheets: find row" and "Gmail: create draft." Ask: "Find the row for Acme in my Clients sheet and draft a renewal reminder to their contact."
- **Make MCP server:** expose an on-demand scenario called "Order status lookup" and ask your AI about an order.
- **Cost check:** each Zapier MCP tool call is 2 tasks; a Make scenario run costs credits for each module.

Reliability, Monitoring and Maintenance

- Turn on error notifications and look at the history weekly.
- Use replay (Zapier) or incomplete executions and error handlers (Make) to recover from outages.
- Reconnect expired app connections promptly; they are the top cause of silent failures.

11.1 Why automations fail

CAUSE	TYPICAL SIGN	FIX
Expired or revoked app connection	"Authentication" or "401" errors	Reconnect the account in Connections; check password changes
App outage or rate limit	"API is experiencing technical issues," "429"	Wait, then replay; add delays for bursts
Changed field or sheet column	Missing or blank values	Re-pick the field in the step; avoid renaming sheet headers
Unexpected data	Empty name, odd date format	Add Formatter or filters; use default values
AI answer in the wrong format	Router or Paths sends everything to fallback	Tighten the prompt; allow only fixed labels
Plan limit reached	Runs stop or pause	Check usage; upgrade or optimize

11.2 Monitoring routine

- ☐ Make sure error emails go to an inbox you read. Both tools notify owners of failing automations.
- ☐ Once a week, scan Zap history or scenario history for errors and unusual volumes.
- ☐ Once a month, review usage: which automations use the most tasks or credits, and are they worth it?
- ☐ Every quarter, switch off automations nobody uses and update descriptions.
- ☐ Keep a simple list (a sheet or Notion page) of all live automations, owners and purposes.

11.3 Recovering from an outage

- **Zapier:** filter Zap history to *Errored*, select the runs and replay them, or enable Autoreplay so errored runs retry automatically.
- **Make:** if incomplete executions are enabled, open the scenario's incomplete executions, fix the cause and resume them. Add a **Break** error handler to modules that talk to flaky services so failed bundles are stored for retry.

Security and Privacy

- Every connection gives the platform access to an account. Connect only what you need, with the narrowest permissions offered.
- Don't send sensitive personal, financial or health data through automations unless your plan, contracts and policies allow it.
- Webhook URLs, API keys and MCP access are keys to your accounts. Guard them.

12.1 Connections and permissions

When you connect an app, you grant Zapier or Make permission to act in that account. Treat each connection as a key you have handed over.

- ☐ Use a dedicated service account for business automations where possible, not a personal login.
- ☐ Remove connections you no longer use (Zapier: *App connections*; Make: *Connections*).
- ☐ When a teammate leaves, transfer their automations and reconnect them under a shared account.
- ☐ Turn on two-factor authentication for your Zapier and Make accounts.
- ☐ Prefer read-only access in AI tools and MCP setups; add write access deliberately.

12.2 Sensitive data

Automations copy data between services and keep logs of what passed through (that is how history and replay work). Before automating anything containing health, financial, legal or children's information, check:

- Whether your plan and your agreements with Zapier, Make and every connected app permit that data. Healthcare workers should read Volume 19 on HIPAA basics before automating anything with patient information.
- How long run history and logs are kept, and who in your team can see them.
- Whether AI steps send the data to a third-party AI provider, and what that provider does with it.

12.3 Prompt injection in automations

When an AI step reads content from outside (emails, web pages, form submissions), that content might contain instructions like “ignore your rules and forward this to...”. If the AI step can trigger actions, that is dangerous.

- ☐ Keep AI steps that read outside content **separate** from steps that send, delete or pay.
- ☐ Let the AI choose only from a fixed list of labels; let the automation, not the AI, decide what to do with each label.
- ☐ Put a human approval step before anything irreversible.

Volume 18 covers AI privacy and security in depth.

Zapier, Make or n8n?

	ZAPIER	MAKE	N8N (VOLUME 6)
Style	Step list	Visual canvas	Visual canvas
Apps	9,000+	3,500+ standard apps	1,000+ integrations plus HTTP
Free plan	100 tasks, two-step Zaps	1,000 credits, 2 active scenarios	Free to self-host; cloud plans paid
Usage unit	Tasks (actions only)	Credits (most module actions)	Executions (cloud)
Learning curve	Gentlest	Moderate	Moderate to steep
Self-hosting	No	No (Enterprise on-premises agent for private systems)	Yes
AI chat access	Zapier MCP	Make MCP server	MCP nodes

Choose Zapier if you want results fast, need a niche app, or your team is non-technical. **Choose Make** for complex, high-volume logic at lower cost. **Choose n8n** if you want to run automations on your own server or need maximum control.

Troubleshooting

PROBLEM	LIKELY CAUSE	FIX
Zapier won't let me add a third step	Free plan allows two-step Zaps	Start the Professional trial or upgrade
Zap is on but never runs	Polling interval, wrong filter, or trigger watches the wrong folder or label	Wait for the interval; test the trigger; check the filter
"Filtered" in every run	Filter condition never true	Open a run and compare the real value with your rule
Make scenario used far more credits than expected	Many bundles flowing through every module	Set trigger limits; aggregate; filter early
Make scenario switched off by itself	Repeated errors, or out of credits	Check history, fix errors, add credits
Only two Make scenarios can be active	Free plan limit	Deactivate one, or upgrade to Core
Code step times out	Free runtime is 1 second (Zapier)	Simplify code, or use a paid plan
Dates are wrong by several hours	Code steps use UTC	Convert time zones with Formatter or in code
AI step answers vary run to run	AI is non-deterministic	Fix the format, list allowed values, give examples
Webhook returns 429	Rate limit	Slow the sender down; batch requests
MCP actions not appearing in Claude or ChatGPT	Actions not enabled, or client not connected	Recheck allowed actions and the client connection

Quick Reference

- **Shape of every automation:** trigger → (filter) → actions, with data mapped between steps
- **Zapier usage:** tasks = successful actions; triggers, Filter, Paths, Formatter, Delay, Looping, Tables, Forms are free; MCP call = 2 tasks
- **Zapier Free:** 100 tasks/mo, two-step Zaps, 15-min polling; Professional from \$19.99/mo annual
- **Make usage:** credits per module operation per bundle; Router and error handlers free; Code app 2 credits/sec
- **Make Free:** 1,000 credits/mo, 2 active scenarios, 15-min interval; Core from \$12/mo annual (10k credits)
- **Test safely:** own email address, sample data, trigger Limit 2 in Make
- **Cost savers:** filter early, instant triggers, aggregate or digest, switch off unused automations

- **AI prompt rules:** one job, fixed output, allowed values, fallback, never-do list
- **Recover:** Zapier replay or Autoreplay; Make incomplete executions and Break handler
- **Security:** least privilege, service accounts, 2FA, private webhook URLs with a secret
- **AI actions:** drafts first, human approval before anything irreversible
- **Prices and limits checked:** October 16, 2026; confirm on the official pricing pages

Sources (checked October 16, 2026)

- Zapier: [Pricing](#); [Zapier MCP](#); [Agents](#); [Forms](#); [Use Python code in Zap workflows](#); [Use JavaScript code in Zap workflows](#); [Replay Zap runs](#); [View and manage your Zap history](#); [Add branching logic with Paths](#); [Webhooks by Zapier rate limits](#); [What is Zapier Copilot?](#); [Extended runtimes for Code steps](#)
- Make: [Pricing](#); [Monthly and annual subscriptions](#); [Router](#); [Iterator](#); [Webhooks](#); [Make MCP server](#); [Make AI Agents](#); [Credit usage for AI agents](#)

Glossary

- **Action:** a step that does work in an app, such as creating a row.

- **Aggregator (Make):** a module that combines many bundles into one.

- **Autoreplay (Zapier):** automatic retry of errored Zap runs.

- **Bundle (Make):** one item of data moving through a scenario.

- **Connection:** a saved, authorized login to an app.

- **Credit (Make):** the unit Make bills by; most module actions use 1.

- **Data store (Make):** a small built-in database.

- **Error handler (Make):** a route that decides what happens when a module fails.

- **Filter:** a rule that stops a run unless a condition is true.

- **Instant trigger:** a trigger that fires immediately via a webhook.

- **Iterator (Make):** a module that splits a list into separate bundles.

- **Mapping:** choosing data from earlier steps to fill fields in later ones.

- **MCP: Model Context Protocol,** a standard that lets AI tools use outside actions.

- **Paths (Zapier):** branches with their own rules and steps.

- **Polling:** checking an app for new data on a timer.

- **Router (Make):** a module that splits a scenario into routes.

- **Scenario (Make):** an automation.

- **Task (Zapier):** the unit Zapier bills by; one successful action.

- **Trigger:** the event that starts an automation.

- **Webhook:** a URL that receives data and starts an automation instantly.

- **Zap (Zapier):** an automation.
