

The Automation Cookbook — 20 Ready-Made n8n Workflows

Stop paying £50/month for Zapier. Build powerful automations for free.

A practical guide to automating your business, development workflow, and personal productivity with n8n — the open-source automation platform you own and control.

1. Why n8n?

If you've ever used Zapier or Make (formerly Integromat), you know the drill: you build a handful of useful automations, hit the free tier limit, and suddenly you're staring at a £50–£150/month bill for what amounts to gluing a few apps together. It's a racket.

n8n is different. It's an open-source, self-hosted workflow automation tool that gives you unlimited workflows, unlimited executions, and zero per-task fees. You run it on your own server — a £5/month VPS handles thousands of workflows without breaking a sweat.

Here's what makes n8n stand out:

- **Self-hosted & private.** Your data never leaves your infrastructure. No third-party processing. Full GDPR compliance by default.
- **400+ built-in integrations.** Slack, Notion, Google Sheets, Stripe, GitHub, OpenAI — if you use it, n8n probably connects to it.
- **Visual workflow builder.** Drag, drop, connect. No code required for most automations, but you can drop into JavaScript or Python when you need to.
- **Fair-code licensed.** The source is open. You can inspect it, modify it, extend it. The community is active and generous with shared workflows.

- **AI-native.** First-class support for OpenAI, Anthropic, and local LLMs — meaning you can build AI-powered automations without middleware.

The bottom line: n8n gives you enterprise-grade automation for the cost of a cheap VPS. This cookbook shows you exactly how to use it.

2. Getting Started

Install n8n with Docker Compose

The fastest way to get n8n running is Docker. Create a `docker-compose.yml` file:

```
version: "3.8"

services:
  n8n:
    image: n8nio/n8n:latest
    restart: unless-stopped
    ports:
      - "5678:5678"
    environment:
      - N8N_BASIC_AUTH_ACTIVE=true
      - N8N_BASIC_AUTH_USER=admin
      - N8N_BASIC_AUTH_PASSWORD=your-strong-password
      - GENERIC_TIMEZONE=Europe/London
      - N8N_SECURE_COOKIE=false
    volumes:
      - n8n_data:/home/node/.n8n

volumes:
  n8n_data
```

Run `docker compose up -d` and visit `http://your-server:5678`. That's it — you're running.

Key Concepts

Before diving into the workflows, understand these building blocks:

- **Trigger nodes** start a workflow. They fire on a schedule (Cron), on a webhook call, when an email arrives, or when an event happens in a connected app.
- **Action nodes** do things: send messages, create records, call APIs, transform data.
- **Function/Code nodes** let you write JavaScript to transform data however you like.
- **IF/Switch nodes** add conditional logic — route data down different paths based on rules.
- **Credentials** are stored encrypted in n8n. Set them up once, reuse across workflows.

Each workflow below lists the trigger, the key nodes in sequence, and enough detail for you to rebuild it from scratch. No JSON dumps — just clear, practical instructions.

3. The 20 Workflows

Content & Marketing

Workflow 1: RSS → AI Summary → Slack

What it does: Monitors RSS feeds from industry blogs, generates a concise AI summary of each new article, and posts it to a Slack channel — giving your team a curated news digest without anyone manually reading dozens of posts.

Complexity: □

Trigger: `Schedule Trigger` — runs every 2 hours (or your preferred interval).

Node sequence:

1. **RSS Feed Read** — Configure with up to 10 feed URLs (e.g., TechCrunch, Hacker News, your niche blogs). Set "Limit" to 5 per feed to avoid flooding.
2. **IF node** — Check

```
new Date(item.pubDate) > new Date(Date.now() - 2 * 60 * 60 * 1000)
```

to filter only articles published since the last run.
3. **HTTP Request** — Fetch the full article content from each URL. Set "Response Format" to "Text" for HTML.
4. **HTML Extract** — Pull the article body text using CSS selector (typically `article` or `.post-content`).
5. **OpenAI node** — Send the extracted text with the prompt: *"Summarise this article in 3 bullet points. Keep it under 80 words. Include why it matters for [your industry]."*
6. **Slack node** — Post to your `#industry-news` channel. Format: bold title as a link, followed by the AI summary.

Use case: Marketing teams staying on top of competitor moves; founders monitoring their space without doomscrolling.

Workflow 2: Blog Post → Auto Social Posts

What it does: When you publish a new blog post, this workflow automatically generates and queues tailored posts for Twitter/X, LinkedIn, and Threads — each adapted to the platform's style and character limits.

Complexity: ☐☐

Trigger: `Webhook` — fired by your CMS (WordPress, Ghost, or Webflow) on new post publish.

Node sequence:

1. **Webhook node** — Receives the payload containing the post title, URL, excerpt, and tags.
2. **HTTP Request** — Fetch the full post body from the URL for richer context.

3. **OpenAI node (Twitter)** — Prompt: *"Write a punchy tweet (under 260 chars) promoting this blog post. Include a hook, the key takeaway, and the URL. No hashtag spam — max 2 relevant hashtags."*
4. **OpenAI node (LinkedIn)** — Prompt: *"Write a professional LinkedIn post (150–250 words) about this article. Open with a thought-provoking question. Include the link at the end."*
5. **OpenAI node (Threads)** — Prompt: *"Write a casual, conversational Threads post (under 400 chars). Be authentic, slightly opinionated."*
6. **Twitter node** — Post the tweet using the Twitter/X API credentials.
7. **LinkedIn node** — Publish using the LinkedIn OAuth credentials.
8. **HTTP Request** — POST to Threads API (or use the community Threads node if available).
9. **Google Sheets node** — Log each post (platform, content, URL, timestamp) to a tracking spreadsheet.

Use case: Solo founders and small marketing teams who want consistent social presence without the manual overhead.

Workflow 3: Content Calendar Reminders

What it does: Reads your content calendar from a Google Sheet or Notion database and sends Slack reminders to the assigned writer 48 hours before the due date, then a final nudge on the day.

Complexity: ☐

Trigger: `Schedule Trigger` — runs daily at 09:00.

Node sequence:

1. **Google Sheets node (or Notion node)** — Read all rows from your content calendar. Expected columns: Title, Assigned To, Due Date, Status, Slack Handle.
2. **IF node (48h warning)** — Filter rows where Due Date is exactly 2 days from now AND Status is not "Published" or "Complete".

3. **Slack node** — Send a direct message: *"Hey @writer — your piece '[Title]' is due in 2 days. Need anything?"*
4. **IF node (due today)** — Filter rows where Due Date is today AND Status is not "Published".
5. **Slack node** — Send to channel: *"📅 Due today: '[Title]' — assigned to @writer."*

Use case: Content teams managing editorial calendars without paying for dedicated project management tools.

Workflow 4: Competitor Price Monitor

What it does: Scrapes competitor pricing pages daily, detects changes, and alerts you instantly — so you're never blindsided by a pricing shift.

Complexity: ☐☐☐

Trigger: `Schedule Trigger` — runs daily at 07:00.

Node sequence:

1. **HTTP Request** — Fetch each competitor's pricing page. Configure one request per competitor (use multiple parallel branches or a loop with the **Split In Batches** node).
2. **HTML Extract** — Pull pricing data using CSS selectors specific to each site (e.g., `.price-value`, `[data-plan="pro"] .amount`). You'll need to inspect each page once to find the right selectors.
3. **Google Sheets node (Read)** — Fetch the previously stored prices from your tracking sheet.
4. **Code node** — Compare current prices against stored values. Flag any differences. Calculate percentage change.
5. **IF node** — Only proceed if a price change was detected.
6. **Google Sheets node (Update)** — Write the new prices and timestamp.
7. **Slack node** — Alert: *"📅 [Competitor] changed their Pro plan from £49 to £39/mo (~20%). [Link to page]"*
8. **Email node (optional)** — CC the pricing alert to your product lead.

Use case: SaaS companies, e-commerce businesses, and agencies tracking competitor pricing strategy.

Workflow 5: SEO Rank Tracker

What it does: Checks your Google search rankings for target keywords daily, logs the positions to a spreadsheet, and alerts you when rankings drop or improve significantly.

Complexity: ☐☐

Trigger: `Schedule Trigger` — runs daily at 06:00 (before your team starts).

Node sequence:

1. **Google Sheets node (Read)** — Fetch your list of target keywords and their last known positions from a tracking sheet.
2. **Split In Batches** — Process 5 keywords at a time (to respect API rate limits).
3. **HTTP Request** — Call a SERP API (e.g., SerpAPI, ValueSERP, or SERPHouse — most have free tiers). Send each keyword as a query, set location to your target market.
4. **Code node** — Parse the API response. Find your domain in the results. Extract position, URL, and featured snippet status.
5. **Code node** — Compare current position with yesterday's. Calculate the delta.
6. **Google Sheets node (Append)** — Log: keyword, position, delta, date, URL found.
7. **IF node** — Filter for significant changes (position moved ± 5 or more, or entered/left top 10).
8. **Slack node** — Weekly summary or instant alerts: "📌 'best project management tool' moved from #14 → #8. 📌 'free CRM software' dropped from #5 → #12."

Use case: SEO professionals and content marketers tracking keyword performance without expensive tools like Ahrefs or SEMrush.

Sales & CRM

Workflow 6: Form Submission → Enrich → CRM

What it does: When someone fills in your website form, this workflow enriches the lead with company data (using Clearbit, Apollo, or a similar service), scores them, and creates a contact in your CRM — ready for your sales team.

Complexity: ☐☐

Trigger: Webhook — called by your form provider (Typeform, Tally, or a simple HTML form).

Node sequence:

1. **Webhook node** — Receives name, email, company, and message.
2. **HTTP Request (Enrichment)** — Call the Clearbit/Apollo API with the email domain. Returns company size, industry, location, estimated revenue, and social profiles.
3. **Code node** — Merge the form data with the enrichment data. Normalise fields (e.g., map company size ranges to T-shirt sizes: S/M/L/Enterprise).
4. **IF node** — Route based on company size: Enterprise goes to the sales team channel, smaller leads go to a nurture sequence.
5. **HubSpot node** (or Pipedrive/Salesforce) — Create or update the contact with all enriched fields.
6. **Slack node** — Notify `#sales-leads`: "👋 New lead: Jane Smith, CTO at Acme Corp (Enterprise, SaaS). Message: 'Interested in API integration.' [CRM link]"

Use case: B2B companies wanting instant lead enrichment without paying for an expensive sales automation platform.

Workflow 7: Stripe Payment → Thank You Email → Slack

What it does: When a customer completes a Stripe payment, automatically sends a personalised thank-you email and notifies your team in Slack — ensuring no sale goes unacknowledged.

Complexity: ▢

Trigger: Stripe Trigger — event type: `checkout.session.completed`.

Node sequence:

1. **Stripe Trigger** — Fires when a payment succeeds. Provides customer email, name, amount, and product details.
2. **Code node** — Format the currency, generate a receipt summary, and set the appropriate email template variables.
3. **Send Email node (or Mailgun/Resend node)** — Send a personalised thank-you email: customer name, what they bought, receipt amount, and a link to access their purchase or account.
4. **Slack node** — Post to `#sales`: *"🎉 New sale! [Customer Name] purchased [Product] for £49. Total revenue today: £X."*
5. **Google Sheets node (optional)** — Append the transaction to a revenue tracking sheet.

Use case: Digital product sellers, SaaS companies, and course creators wanting instant post-purchase engagement.

Workflow 8: Lead Scoring Pipeline

What it does: Assigns a numerical score to every new lead based on configurable criteria (company size, job title, engagement level), then routes high-scoring leads to immediate follow-up and low-scoring ones to nurture campaigns.

Complexity: ▢▢▢

Trigger: Webhook — fired when a new lead enters your CRM or form pipeline.

Node sequence:

1. **Webhook node** — Receives lead data: name, email, company, job title, source, pages visited.
2. **HTTP Request** — Enrich with company data (Clearbit/Apollo) if not already enriched.
3. **Code node (Scoring)** — Apply your scoring rules in JavaScript:
 4. Job title contains "CEO/CTO/VP/Director": +30 points
 5. Company size >100 employees: +20 points
 6. Visited pricing page: +25 points
 7. Came from paid ad: +10 points
 8. Free email domain (gmail, yahoo): -15 points
 9. Total = sum of all applicable scores
10. **Switch node** — Route based on score:
11. **Hot (70+)**: Immediate sales outreach path.
12. **Warm (40–69)**: Nurture sequence path.
13. **Cold (<40)**: Add to general mailing list.
14. **HubSpot node** — Update the contact record with the lead score and segment tag.
15. **Slack node (Hot leads only)** — Instant alert: "🔥 *Hot lead (score: 85): Sarah Chen, VP Engineering at DataCorp. [CRM link]*"

Use case: Sales teams prioritising outreach based on data rather than gut feeling.

Workflow 9: Follow-Up Auto-Reminder

What it does: Monitors your CRM for leads that haven't been contacted within a set period and sends escalating reminders to the assigned sales rep — and eventually their manager.

Complexity: ☐☐

Trigger: `Schedule Trigger` — runs every morning at 08:30.

Node sequence:

1. **HubSpot node** (or your CRM) — Fetch all leads with status "New" or "In Progress" and their last activity date.
2. **Code node** — Calculate days since last contact for each lead. Flag those overdue: >2 days = gentle nudge, >5 days = urgent, >7 days = escalate to manager.
3. **Switch node** — Route by urgency level.
4. **Slack node (Gentle)** — DM the rep: *"Friendly reminder: you haven't followed up with [Lead Name] in 3 days. [CRM link]"*
5. **Slack node (Urgent)** — DM the rep + post in #sales : "⚠ [Lead Name] hasn't been contacted in 6 days. Please follow up today."
6. **Slack node (Escalate)** — DM the sales manager: "🔔 [Lead Name] assigned to [Rep] hasn't been contacted in 8 days. Needs attention."

Use case: Sales teams ensuring no lead falls through the cracks.

Workflow 10: Invoice Generator

What it does: Generates a professional PDF invoice from a Google Sheets row or form submission, emails it to the client, and logs it to your accounting spreadsheet.

Complexity: ☐☐

Trigger: Webhook — or **Google Sheets Trigger** when a new row is added to your "Invoices to Send" sheet.

Node sequence:

1. **Google Sheets node** — Read the invoice details: client name, address, line items, amounts, VAT rate, due date, your payment details.
2. **Code node** — Calculate subtotal, VAT amount, and total. Generate a unique invoice number (e.g., INV-2025-0042). Format dates and currency.
3. **HTML node / Code node** — Build an HTML invoice template. Use a clean, professional layout with your logo, line items table, totals, and payment terms. Inline CSS for email compatibility.

4. **HTTP Request** — Convert the HTML to PDF using a free API (e.g., `html2pdf.app`, or self-hosted `Gotenberg`). Alternatively, use the **Gotenberg node** if available.
5. **Send Email node** — Email the PDF as an attachment to the client with a polite message: *"Please find attached invoice [number] for £X, due by [date]."*
6. **Google Sheets node** — Log the invoice: number, client, amount, date sent, status "Sent".
7. **Google Drive node (optional)** — Save the PDF to a `Invoices/2025/` folder for your records.

Use case: Freelancers, agencies, and small businesses generating invoices without paying for dedicated invoicing software.

DevOps

Workflow 11: Uptime Check → Alert

What it does: Pings your websites and APIs at regular intervals and fires alerts to Slack, email, or PagerDuty the moment something goes down — plus a recovery notification when it comes back.

Complexity: □

Trigger: `Schedule Trigger` — runs every 5 minutes.

Node sequence:

1. **Code node** — Define your endpoints as an array: `[{name: "Main Site", url: "https://example.com", expectedStatus: 200}, ...]`.
2. **Split In Batches** — Process each endpoint.
3. **HTTP Request** — GET each URL. Set timeout to 10 seconds. Enable "Never Error" so the workflow continues even on failures.
4. **IF node** — Check if the response status code matches the expected value. Also check if the response time exceeded a threshold (e.g., 3 seconds).

5. **IF node (State Check)** — Compare against the last known status (stored in a Google Sheet or n8n's built-in static data). Only alert on *changes* to avoid repeated notifications.
6. **Slack node** — Down alert: "🔔 [Service Name] is DOWN. Status: 503. Response time: timeout. Checked at 14:32 UTC." Recovery: "🔔 [Service Name] is back UP. Downtime: ~15 minutes."
7. **Google Sheets node** — Log uptime history for reporting.

Use case: Any team running web services that needs basic uptime monitoring without paying for Pingdom or UptimeRobot Pro.

Workflow 12: GitHub Issue → Auto-Classify → Assign

What it does: When a new GitHub issue is opened, uses AI to classify it (bug, feature request, question, documentation), applies the appropriate label, and assigns it to the right team member based on the content.

Complexity: ☐☐

Trigger: `GitHub Trigger` — event: "Issues" → action: "opened".

Node sequence:

1. **GitHub Trigger** — Receives the issue title, body, author, and repository.
2. **OpenAI node** — Prompt: "Classify this GitHub issue into one category: bug, feature-request, question, documentation. Also identify the component affected (frontend, backend, API, infrastructure, other). Issue title: '[title]'. Body: '[body]'. Respond in JSON: {category, component, priority, summary}."
3. **Code node** — Parse the AI response. Map the component to an assignee using a lookup table (e.g., frontend → @alice, backend → @bob).
4. **GitHub node (Add Labels)** — Apply labels matching the category and component (e.g., `bug`, `backend`).
5. **GitHub node (Assign)** — Assign the issue to the appropriate team member.
6. **GitHub node (Comment)** — Post a comment: "🔔 Auto-classified as [category] ([component]). Assigned to @[assignee]. Priority: [priority]."

Use case: Open-source maintainers and development teams drowning in unorganised issues.

Workflow 13: Backup Verification → Report

What it does: Checks that your automated backups actually ran and produced valid, recent files — then sends a daily verification report. Catches silent backup failures before they become disasters.

Complexity: ☐☐

Trigger: `Schedule Trigger` — runs daily at 04:00 (after backups complete).

Node sequence:

1. **SSH node** (or **Execute Command** node) — Connect to your backup server. List backup files in the target directory: `ls -la /backups/daily/ | tail -5`. Capture file names, sizes, and modification dates.
2. **Code node** — Parse the file listing. Check: (a) Does today's backup file exist? (b) Is the file size reasonable (>1 MB, or within 20% of the previous day's file)? (c) Was it modified within the last 24 hours?
3. **SSH node (optional)** — For database backups, test the integrity: `gzip -t /backups/daily/db-2025-01-15.sql.gz` or attempt a test restore to a scratch database.
4. **IF node** — Route based on pass/fail.
5. **Slack node (Pass)** — "👍 Backup verification passed. DB: 2.4 GB (±3% vs yesterday). Files: 890 MB. All checksums valid."
6. **Slack node (Fail)** — "🔴 BACKUP FAILURE: [details]. Last valid backup: [date]. Immediate investigation required."
7. **Email node (Fail only)** — Send a high-priority email to the on-call engineer.

Use case: Any team relying on automated backups who's been burnt by silent failures (i.e., everyone, eventually).

Workflow 14: SSL Certificate Expiry Checker

What it does: Checks your SSL certificates daily and warns you 30, 14, and 7 days before they expire — preventing those embarrassing "Your connection is not secure" moments.

Complexity: ▢

Trigger: `Schedule Trigger` — runs daily at 08:00.

Node sequence:

1. **Code node** — Define your domains: `["example.com", "api.example.com", "app.example.com"]`.
2. **Code node (SSL Check)** — For each domain, use n8n's built-in HTTP request capabilities or a shell command `(echo | openssl s_client -servername [domain] -connect [domain]:443 2>/dev/null | openssl x509 -noout -enddate)` via the **Execute Command** node. Parse the expiry date.
3. **Code node** — Calculate days remaining for each certificate. Flag any under 30 days.
4. **Switch node** — Route by urgency: >30 days (ignore), 14–30 days (info), 7–14 days (warning), <7 days (critical).
5. **Slack node** — Graduated alerts:
6. Info: *"📅 SSL cert for api.example.com expires in 21 days."*
7. Warning: *"⚠️ SSL cert for app.example.com expires in 9 days. Renew soon."*
8. Critical: *"🔴 SSL cert for example.com expires in 3 days! Immediate action needed."*

Use case: DevOps teams and anyone managing multiple domains — especially those not using auto-renewal (or wanting to verify auto-renewal works).

Workflow 15: Docker Container Health → Restart → Log

What it does: Monitors Docker container health status, automatically restarts unhealthy containers, and logs all incidents — giving you self-healing infrastructure with full audit trails.

Complexity: ☐☐☐

Trigger: `Schedule Trigger` — runs every 3 minutes.

Node sequence:

1. **Execute Command node** — Run `docker ps --format '{{.Names}}\t{{.Status}}\t{{.State}}' --filter "health=unhealthy"` to list any unhealthy containers.
2. **IF node** — Check if the output is non-empty (unhealthy containers found).
3. **Code node** — Parse the container names and status information from the command output.
4. **Execute Command node** — For each unhealthy container: `docker restart [container_name]`.
5. **Wait node** — Pause for 30 seconds to allow the container to start.
6. **Execute Command node** — Check the restarted container's status: `docker inspect --format '{{.State.Health.Status}}' [container_name]`.
7. **IF node** — Did the restart fix the issue?
8. **Slack node (Recovered)** — "`👍 Container [name] was unhealthy and has been auto-restarted. Status: healthy.`"
9. **Slack node (Still failing)** — "`👎 Container [name] is still unhealthy after restart. Manual intervention needed. Last 20 log lines: [logs].`"
10. **Google Sheets node** — Log every incident: container name, timestamp, action taken, outcome.

Use case: Small DevOps teams running Docker in production who want basic self-healing without Kubernetes.

Personal Productivity

Workflow 16: Daily Digest Email

What it does: Every morning, compiles a personalised digest — today's calendar events, weather forecast, top news headlines, and your task list — into a single beautifully formatted email.

Complexity: ☐☐

Trigger: `Schedule Trigger` — runs daily at 06:30.

Node sequence:

1. **Google Calendar node** — Fetch today's events. Format as a tidy list with times and meeting links.
2. **HTTP Request (Weather)** — Call OpenWeatherMap API for your location. Extract temperature, conditions, and rain probability.
3. **HTTP Request (News)** — Call a news API (NewsAPI or GNews) for your chosen categories. Get the top 5 headlines with URLs.
4. **Todoist node** (or Notion/Google Tasks) — Fetch tasks due today.
5. **Code node** — Compile everything into an HTML email template. Use sections: "☼ Weather", "📅 Today's Schedule", "📰 Headlines", "📋 Tasks". Style it clean and mobile-friendly with inline CSS.
6. **Send Email node** — Send to yourself. Subject: *"Your Daily Briefing — [Day, Date]"*.

Use case: Anyone who wants to start their morning with everything they need in one place, without checking five different apps.

Workflow 17: Telegram → Notion Bookmark Saver

What it does: Send yourself a URL via Telegram and it's automatically saved to your Notion bookmarks database — with the page title, description, and tags extracted automatically.

Complexity: ▯

Trigger: Telegram Trigger — listens for messages sent to your bot.

Node sequence:

1. **Telegram Trigger** — Receives the message. Extract any URL using a regex in a Code node.
2. **HTTP Request** — Fetch the URL to grab the page's HTML.
3. **HTML Extract** — Pull the `<title>` tag, meta description, and Open Graph image.
4. **OpenAI node (optional)** — Prompt: *"Based on this page title and description, suggest 2–3 tags from this list: [tech, business, design, productivity, reading, reference, tutorial]."*
5. **Notion node** — Create a new page in your Bookmarks database with properties: Title, URL, Description, Tags, Date Saved, Status (default: "Unread").
6. **Telegram node** — Reply: "✅ Saved: '[Page Title]' to Notion. Tagged: #tech #tutorial."

Use case: Avid readers and researchers who want a quick "save for later" workflow without context-switching.

Workflow 18: Email Attachments → Auto-File

What it does: Monitors your inbox for emails with attachments and automatically files them into the correct Google Drive folder based on the sender, file type, or subject line keywords.

Complexity: ▯▯

Trigger: Email Trigger (IMAP) — checks for new emails with attachments every 10 minutes.

Node sequence:

1. **Email Trigger** — Configured with your IMAP credentials. Filter for emails with attachments.

2. **Code node** — Extract sender domain, subject line, and attachment details (filename, MIME type, size).
3. **Switch node** — Route based on rules:
 4. Sender domain is `invoices@supplier.com` or subject contains "invoice" → `Finance/Invoices/` folder.
 5. Attachment is a PDF from a known client → `Clients/[Client Name]/` folder.
 6. Attachment is an image → `Media/Screenshots/` folder.
 7. Everything else → `Inbox/Unsorted/` folder.
8. **Google Drive node** — Upload the attachment to the determined folder. Name the file with a date prefix: `2025-01-15_original-filename.pdf`.
9. **Google Sheets node** — Log the filing: original email subject, sender, filename, destination folder, date.
10. **Email node (optional)** — Send yourself a daily summary: "📧 Auto-filed 7 attachments today."

Use case: Anyone drowning in email attachments — particularly freelancers and small business owners receiving invoices, contracts, and assets from multiple sources.

Workflow 19: Weekly Time Tracking Summary

What it does: Pulls your time tracking data from Toggl (or Clockify/Harvest), generates a breakdown by project and category, and sends you a weekly report with insights.

Complexity: ☐☐

Trigger: `Schedule Trigger` — runs every Friday at 17:00.

Node sequence:

1. **HTTP Request** — Call the Toggl API (or Clockify) for this week's time entries. Use the detailed report endpoint with start/end date parameters.
2. **Code node** — Aggregate the data:
3. Total hours worked.

4. Hours per project.
5. Hours per category (deep work, meetings, admin, etc.).
6. Billable vs. non-billable split.
7. Busiest and quietest days.
8. **Code node** — Generate a formatted summary with ASCII bar charts or simple tables.
9. **OpenAI node (optional)** — Prompt: *"Here's my time breakdown this week: [data]. Give me 2 sentences of practical advice on improving my time allocation."*
10. **Send Email node** — Send the weekly report to yourself. Subject: "📅 Week of [date]: [total]h tracked".
11. **Google Sheets node** — Append weekly totals to a running spreadsheet for month-over-month comparison.

Use case: Freelancers tracking billable hours, or anyone who wants accountability over how they spend their work week.

Workflow 20: Meeting Audio → Transcription → Summary

What it does: Takes a meeting recording (uploaded or from a cloud storage link), transcribes it using AI, generates structured meeting notes with action items, and distributes them to attendees.

Complexity: 🟢🟢🟢

Trigger: `Webhook` — or **Google Drive Trigger** when a new audio/video file is added to a specific folder.

Node sequence:

1. **Google Drive Trigger** — Watches a "Meeting Recordings" folder for new files (MP3, MP4, M4A, WAV).
2. **Google Drive node** — Download the file binary.
3. **HTTP Request** — Send the audio to a transcription API. Use **OpenAI's Whisper API** (or AssemblyAI for speaker diarisation). For files over 25 MB, use the Code node to split into chunks first.

4. **OpenAI node** — Send the transcript with the prompt: *"Create structured meeting notes from this transcript. Include: (1) Meeting summary (3–5 sentences), (2) Key decisions made, (3) Action items with assigned owners if mentioned, (4) Open questions. Format in Markdown."*
5. **Notion node** — Create a new page in a "Meeting Notes" database. Properties: Meeting Date, Title (parsed from filename or transcript), Attendees, Summary, and the full notes as page content.
6. **Slack node** — Post to the relevant channel: *"📌 Meeting notes for [Title] are ready: [Notion link]. Action items: [list]."*
7. **Send Email node** — Email a copy of the notes to all attendees (from a participant list in the trigger data or a lookup sheet).

Use case: Teams who record meetings but never get round to writing up notes. This ensures every meeting produces actionable documentation automatically.

4. Advanced Tips

Error Handling

Every workflow should have error handling. In n8n, add an **Error Trigger** workflow that catches failures from any workflow, logs them, and sends a Slack or email notification. Within individual workflows, use the "Continue On Fail" option on HTTP Request nodes and wrap risky operations in **Try/Catch** patterns using the IF node to check for error outputs.

Credentials Management

Never hardcode API keys. Use n8n's built-in credential store — it encrypts everything at rest. For team environments, set up credential sharing so team members can use integrations without seeing the actual keys. Rotate keys quarterly, and use least-privilege scoping (read-only where possible).

Scaling & Performance

For high-volume workflows, switch from SQLite (the default) to **PostgreSQL** as n8n's database — it handles concurrent executions far better. Enable **queue mode** with Redis for parallel processing. Set execution timeouts to prevent runaway workflows. Use the **Split In Batches** node when processing large datasets to avoid memory issues. For self-hosted deployments, monitor n8n's memory usage and set appropriate container limits.

Testing

Use n8n's manual execution mode to test workflows step by step. Pin test data to nodes during development so you don't need to trigger real events repeatedly. Keep a "Sandbox" folder for experimental workflows, separate from production ones.

5. Appendix

Appendix A: n8n vs Zapier vs Make — Comparison

Feature	n8n (Self-hosted)	Zapier	Make
Price (comparable tier)	Free + £5/mo VPS	£56/mo (Starter)	£9/mo (Core)
Workflows	Unlimited	20 (Starter)	Unlimited
Executions/ month	Unlimited	750	10,000
Self-hosted option	☑ Yes	☐ No	☐ No
Open source	☑ Fair-code	☐ No	☐ No
Built-in AI nodes	☑ OpenAI, Anthropic, local LLMs	☐ Limited	☐ Limited

Feature	n8n (Self-hosted)	Zapier	Make
Code nodes	▯ JavaScript & Python	▯ JavaScript & Python	▯ Limited scripting
Branching/logic	▯ IF, Switch, Merge	▯ Paths, Filters	▯ Routers, Filters
Community nodes	▯ 100+ community packages	▯ No	▯ No
Data residency	▯ Your server, your rules	▯ US/EU only	▯ EU/US
Multi-step workflows	▯ Unlimited steps	⚠ 2 steps on Free	▯ Unlimited
Webhook support	▯ Free	⚠ Paid plans only	▯ All plans
Learning curve	Moderate	Easy	Moderate

Verdict: If you're comfortable with Docker and want full control, n8n wins on cost and flexibility. If you want zero setup, Zapier is easiest but expensive. Make sits in the middle.

Appendix B: Production Docker Compose

For production deployments with PostgreSQL, proper networking, and automatic restarts:

```
version: "3.8"
|
services:
  n8n:
    image: n8nio/n8n:latest
    restart: always
```

```
ports:
  - "5678:5678"

environment:
  - N8N_BASIC_AUTH_ACTIVE=true
  - N8N_BASIC_AUTH_USER=${N8N_USER}
  - N8N_BASIC_AUTH_PASSWORD=${N8N_PASSWORD}
  - N8N_HOST=${N8N_HOST}
  - N8N_PORT=5678
  - N8N_PROTOCOL=https
  - WEBHOOK_URL=https://${N8N_HOST}/
  - GENERIC_TIMEZONE=Europe/London
  - DB_TYPE=postgresdb
  - DB_POSTGRESDB_HOST=postgres
  - DB_POSTGRESDB_PORT=5432
  - DB_POSTGRESDB_DATABASE=n8n
  - DB_POSTGRESDB_USER=${POSTGRES_USER}
  - DB_POSTGRESDB_PASSWORD=${POSTGRES_PASSWORD}
  - EXECUTIONS_DATA_PRUNE=true
  - EXECUTIONS_DATA_MAX_AGE=168
  - N8N_METRICS=true

volumes:
  - n8n_data:/home/node/.n8n

depends_on:
  postgres:
    condition: service_healthy

networks:
  - n8n-net

postgres:
  image: postgres:16-alpine
  restart: always
  environment:
    - POSTGRES_USER=${POSTGRES_USER}
    - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    - POSTGRES_DB=n8n
```

```

volumes:
  - postgres_data:/var/lib/postgresql/data

healthcheck:
  test: ["CMD-SHELL", "pg_isready -U ${POSTGRES_USER} -d n8n"]
  interval: 10s
  timeout: 5s
  retries: 5

networks:
  - n8n-net

volumes:
  n8n_data
  postgres_data

networks:
  n8n-net

```

Create a `.env` file alongside it:

```

N8N_USER=admin
N8N_PASSWORD=your-strong-password-here
N8N_HOST=n8n.yourdomain.com
POSTGRES_USER=n8n
POSTGRES_PASSWORD=another-strong-password

```

Use a reverse proxy (Caddy, Nginx, or Traefik) in front for HTTPS. Caddy is the simplest — two lines of Caddyfile and you have automatic Let's Encrypt certificates.

Appendix C: Useful Community Nodes

The n8n community publishes additional nodes you can install to extend functionality. Here are the most useful:

- **n8n-nodes-gutenberg** — Convert HTML to PDF within your workflows (perfect for the invoice generator).

- **n8n-nodes-text-manipulation** — Advanced string operations without writing code.
- **n8n-nodes-puppeteer** — Full browser automation for scraping JavaScript-heavy sites.
- **n8n-nodes-quepasa** — WhatsApp integration for messaging workflows.
- **n8n-nodes-cloudinary** — Image manipulation and optimisation.
- **n8n-nodes-reddit** — Reddit API integration for content monitoring.
- **n8n-nodes-evolution-api** — Another WhatsApp integration option.
- **n8n-nodes-discord** — Extended Discord bot capabilities beyond the built-in node.

Install community nodes from n8n's settings panel under "Community Nodes" — just paste the npm package name and click install.

That's your cookbook. Twenty workflows, ready to build. Each one solves a real problem, uses nodes you can configure in minutes, and costs you nothing beyond your server. Pick the ones that match your needs, customise them, and start automating.

The best automation is the one that quietly saves you two hours a week — every week — while you focus on the work that actually matters.

The Automation Cookbook — 20 Ready-Made n8n Workflows Price: £14 |

License: Personal & small-team use