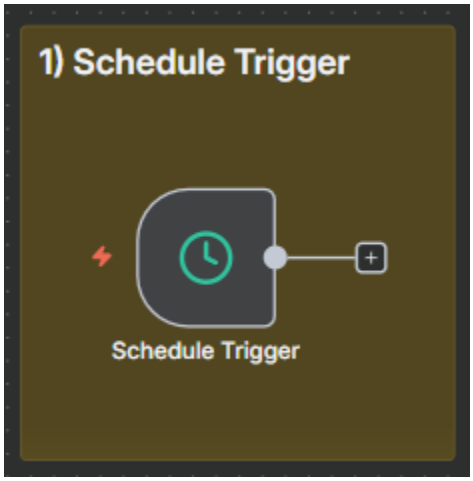
Master n8n with 17 Nodes

By: [Nate Herk]

---

1. The Schedule Trigger



The Schedule Trigger node in n8n is a foundational automation tool that allows workflows to run at fixed times or intervals, making your automations hands-free and scalable, even while you sleep.

What the Schedule Trigger Node Does

The Schedule Trigger node fires a workflow based on a time-based schedule you set, such as every day at 8 a.m., every Monday at 9 a.m., every ten minutes, or any custom pattern using CRON expressions. It works much like the Unix cron utility but is integrated seamlessly with n8n's workflow environment.

Key Use Cases

- Automating data syncs with other platforms (e.g., Google Sheets, Notion) overnight.
- Triggering daily, weekly, or monthly reports, such as sales or analytics dashboards.
- Sending scheduled reminders, notifications, or batch emails.
- Running recurring maintenance or cleanup tasks (e.g., database archiving).

- Powering AI agents or bots to perform regular info retrieval or posting.

How It Works and How To Set Up

1. Add the Node: In your workflow, search for “Schedule Trigger” and add it to your canvas.
2. Configure Intervals: Choose between intervals (minutes, hours, days, weeks) or use a CRON pattern for advanced schedules (such as “every Tuesday at 5 a.m.”).
3. Set Timezone: The node uses the workflow’s timezone or falls back to your n8n instance’s timezone, so check this to ensure accurate timing.
4. Connect the Workflow: Attach the Schedule Trigger node to the rest of your automation nodes (such as HTTP requests, data processing nodes, or email nodes).
5. Activate the Workflow: The workflow must be activated for the trigger to work, you can test it with manual runs first.

Best Practices and Tips

- If you need dynamic schedule logic (such as running based on a value in a database), set a frequent trigger (like every minute) and use a conditional IF node to decide whether to proceed.
- Always double-check your timezone settings to prevent off-schedule execution.
- Keep in mind that excessive frequency (like every minute) may impact instance resources.

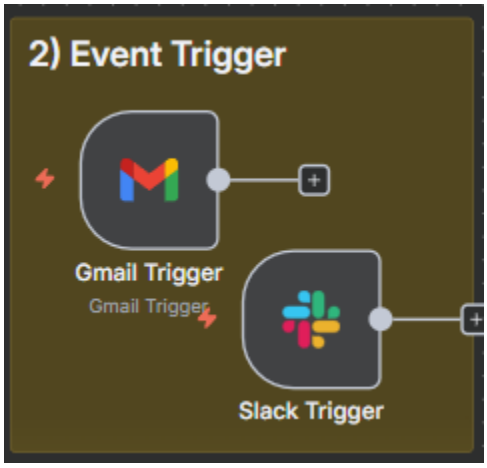
Example Scenario

Suppose you want to run an automation every Monday at 9 a.m.:

- Add the Schedule Trigger node, select a weekly interval, set “1” for weeks between triggers, select “Monday” as the weekday, then choose 9 a.m..
- Connect it to your data-fetch or notification nodes.
- Activate the workflow so it runs automatically each Monday morning.

The Schedule Trigger node is the go-to tool for reliably automating recurring tasks in n8n workflows, allowing your business processes to operate on autopilot.

2. Event Trigger



Event trigger nodes in n8n, such as Gmail Trigger and Slack Trigger, allow your workflows to start automatically whenever specific events happen in an app, making real-time automation possible.

What are Event Trigger Nodes?

These nodes listen for external events, like a new email in Gmail or a new message in Slack, and kick off your workflow immediately when the event occurs. They transform n8n from a tool that just runs on a schedule to one that can powerfully react to your business in real time.

Example: Gmail Trigger Node

- Function: Triggers your workflow when a new message hits your Gmail inbox or matches custom filters.
- When to Use: Set up when you need automations to respond instantly to new emails, like parsing attachments, replying automatically, or logging messages to databases.
- Setup Tips:
    - Authenticate n8n with your Google account.
    - Choose poll frequency, and optionally apply filters for sender, label, unread status, or even Gmail search syntax.
    - Decide if you want a simplified payload (just core fields) or raw data from emails.
    - Can be limited to trigger only on certain labels, from certain senders, or based on specific search criteria.

Example: Slack Trigger Node

- Function: Fires a workflow on events in Slack, like when a message is posted, someone reacts with an emoji, or a new channel/user is created.
- When to Use: Ideal for instant actions such as notification routing, message logging, or workflow initiation based on chat activity.
- Setup Tips:
    - Authenticate with your Slack workspace using the Slack API.

Best Practices and Tips

- Polling vs. Webhooks: Some app triggers (like Gmail) use polling, n8n checks for events every X minutes, while others might use webhooks for instant event notifications.
- Use filters in trigger nodes to avoid triggering on unwanted events and to keep your workflow executions focused.
- Testing event triggers: Activate the workflow and then cause the event (like sending an email or posting in Slack) to verify the trigger setup. Manual executions are for testing, but only “active” mode listens for events automatically.
- For integrations with unsupported events or custom apps, consider using the Webhook Trigger node or generic HTTP triggers.

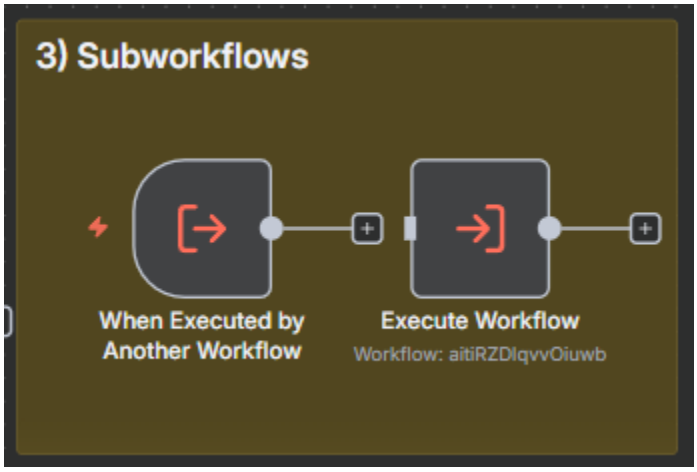
Common Event Trigger Nodes in n8n

Node	**Typical Event Example**	**Key Setup/Notes**
Gmail Trigger	New Email Received	Use filters, polling
Slack Trigger	New Message, Reaction, Channel Created	App credentials needed
HubSpot Trigger	New Contact, Form Submission	Setup app connection

Google Calendar	New Event, Event Update	Select calendar, event
Webhook Trigger	Incoming HTTP Request	Fully custom/webhook

Event triggers are essential for building responsive, "always-on" automations that react instantly to business events across your tools and platforms.

3. Subworkflows



Sub-workflow nodes in n8n (the Execute Sub-workflow node and its trigger) enable workflows to call other workflows modularly, making complex business automations scalable, DRY (don't repeat yourself), and easy to maintain.

What Sub-workflow Nodes Do

- Execute Sub-workflow Node: Lets any workflow invoke another workflow within n8n, passing along structured data as inputs, and receiving the sub-workflow's outputs.
- Sub-workflow Trigger (When Executed by Another Workflow): Sits at the start of the called workflow, specifying how it will handle and process input from the calling workflow.
- This architecture allows you to build core business logic (like "Send Notification" or "Clean Data") once and reuse it in as many workflows as needed.

Key Use Cases

- Sharing utility functions: e.g., centralized error handling, formatting, reporting, or API wrappers.
- Team collaboration: everyone can use and contribute to modular tools instead of copy-pasting nodes between workflows.
- Scaling: update logic in one place and instantly propagate improvements or fixes across dozens of workflows.
- Building agentic systems: sub-workflows can function as tools or actions invoked by AI agents within more complex orchestrations.

How to Set Up Sub-workflow Nodes

Creating the Sub-workflow:

1. Create a new workflow that will act as the sub-workflow.
2. Add the “When Executed by Another Workflow” trigger at the start. Choose the input mode, define fields, a JSON example, or “accept all data” for flexible inputs.
3. Build the workflow as usual, processing the received inputs and generating results.
4. Adjust “This workflow can be called by” in settings to control access for security or collaboration.

Calling the Sub-workflow:

1. In your parent workflow, use the Execute Sub-workflow node.
2. Select “Database” to reference by workflow ID, “Local File” for pre-saved JSON, “Parameter” for direct JSON, or “URL” for remote sources.
3. Map the input data to match what the sub-workflow expects, automatic mapping is available if you declare structured fields.
4. Activate the parent workflow and test the execution. You can view execution history links between parent and sub-workflow for easier debugging.

Best Practices and Tips

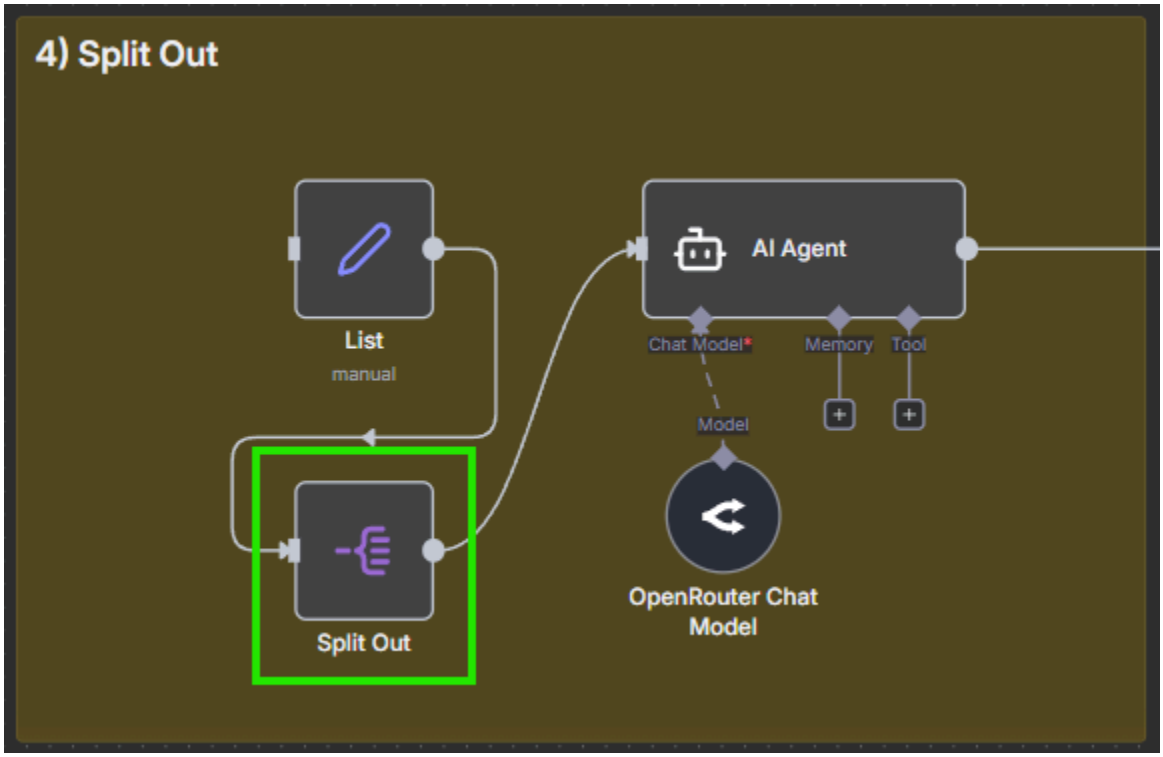
- Each sub-workflow should validate its inputs and handle errors gracefully, especially with “accept all data” mode.
- Pin example input data during development for easier debugging and repeatable tests.
- Minimize main workflow nodes (ideally 4–6), push complex logic into sub-workflows for cleaner architecture.
- Use descriptive names and workflow documentation so other users can easily discover reusable modules.

Example Scenario

- Creating a “Send Slack Report” workflow as a sub-workflow.
- Parent workflows in Marketing, Sales, and Support can each call “Send Slack Report” whenever needed, passing in custom data.

By modularizing with sub-workflows, n8n becomes a platform for building robust, collaborative automation systems, empowering you and your students to scale and maintain complex automations with ease.

4. Split Out



The Split Out node in n8n lets you take any array field in your data and break it down, so each array element becomes its own item for one-by-one processing, just like the “iterator” node in other automation tools.

What the Split Out Node Does

- It receives an item (for example, with a field called “customers”, which is an array: [{"name":"Alice"},{"name":"Bob"},{"name":"Carlos"}]).
- The node produces a separate item for every entry in that array, expanding the single bundled array into multiple workflow items to process individually.
- This means you can send personalized emails to each customer, process separate records from an API response, or handle each row from a spreadsheet in isolation.

Key Use Cases

- Splitting records from API responses so each gets processed separately.
- Breaking out rows from a table, like a Google Sheet or database query, to apply individual actions.

- Iterating over outputs from AI models, so you can treat each generated result individually.
- Fine-grained automation actions, like messaging each contact from a contact list.

How to Set Up and Use the Split Out Node

1. Insert the Node: Add the Split Out node directly after a node that outputs an array (e.g., HTTP Request, Sheets, AI, etc.).
2. Field to Split: Specify the field that holds the array you want to split (e.g., “data.customers”).
3. Include Settings: Decide if you want to keep other fields from the original item with each split-result:
    - “No Other Fields” (just the main field),
    - “All Other Fields” (preserve context),
    - or “Selected Other Fields” (custom list).
4. Dot Notation: By default, you can reference sub-fields using dot notation (“parent.child”). This is helpful for structured data, disable if using flat objects.
5. Connect Downstream: Continue building your workflow, each new item now represents an individual executed path for just that array element.

Best Practices and Tips

- Use Split Out before loops, HTTP requests, or messaging actions, you’ll guarantee that each data piece gets processed without manual array handling.
- If your source arrary comes from deeper in the workflow (not just the previous node), double-check node connections as Split Out expects input directly from its predecessor unless using Merge or Set nodes.
- Combine with Aggregate node if you need to rebuild arrays after processing for reporting, summary, or API pushes.

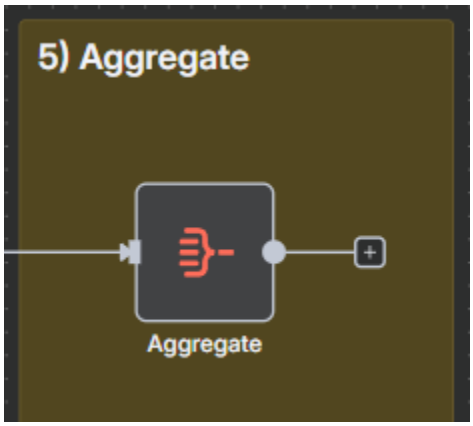
Example Scenario

Suppose an API returns `[{"name":"Julie"},{"name":"Sam"},{"name":"Lee"}]` in a “contacts” field. With Split Out:

- The node produces three separate items for Julie, Sam, and Lee.
- You can then send one email per contact or log each to a database individually.

The Split Out node is critical for shifting automations from “bulk” to “individual”, enabling detailed, targeted, and powerful workflows in n8n.

5. Aggregate



The Aggregate node in n8n is the counterpart to the Split Out node, allowing you to combine (or “bundle up”) multiple workflow items into a single item, often to prep data for summary reports, group processing, or bulk API updates.

What the Aggregate Node Does

- Turns many individual workflow items into a smaller number of grouped items, usually just one, by either collecting data from a field (e.g., all “email” values) or combining entire items into a single array.
- Lets you prepare batch payloads, make summary calculations, or create grouped reports before sending the data onward in your workflow.

Key Use Cases

- Batching for APIs: Prepare arrays of records to send to an external system in bulk, instead of one at a time.
- Summarizing: Compute aggregate metrics (like totals, averages, or min/max) for reporting.
- Grouping: Group items by a specific field (like customer ID or category) to restructure your output for analytics or dashboarding.
- Rebuilding Arrays: After processing individual items (for example, after a Split Out), bundle them back together for further processing or export.

How to Set Up and Use the Aggregate Node

1. Insert the Node: Add the Aggregate node after a series of items, such as outputs from Split Out, HTTP requests, or transformed records.
2. Choose Aggregate Mode:

    - Individual Fields: Combine specific fields from all input items into an array. e.g., collect all email addresses into one array field.
    - All Item Data: Bundle entire items into a list inside a single output field, often named something like “orders” or “results”.
3. Advanced Options:
    - Group By: Group items by a unique value or category (e.g., all sales per customer).
    - Merge Lists: Flatten arrays from the aggregated field if each input is already an array (“list of lists” problem).
    - Include/Exclude Fields: Fine-tune which fields are kept in arrays or summary items.
    - Disable Dot Notation: Reference fields with names containing dots as literal keys (useful with certain database exports).
4. Connect Downstream: After aggregation, output is a single item (or grouped items if using Group By) containing arrays or summarized data, ready for reporting, export, or a bulk action.

Best Practices and Tips

- Use after Split Out or looping logic to recombine individual results back into structure for the next automation step.
- When summarizing, leverage JavaScript nodes downstream for complex calculations (e.g., totals/averages) if not handled in Aggregate directly.
- Always double-check that your resulting array or grouped data matches what the next downstream system or API is expecting.

Example Scenario

Suppose you have 10 order items coming in individually. Using Aggregate:

- All Item Data: Outputs a single item with a “orders” field, value is an array of all your order records, ideal for one-shot API POSTs or summary sheets.
- Individual Fields + Sum: Gathers all “amount” fields into a single array, or sums them up for a total sales value.

The Aggregate node is the critical “recombine” or “grouping” step, helping you move from granular data handling back to bundled, batch-friendly data for efficient automations.

6. Set “Edit Fields”

The Set node, now called the Edit Fields node in n8n, is the cornerstone for shaping and restructuring data as your “source of truth.” It lets you create, rename, format, combine, and prepare any variable your workflow needs for downstream nodes.

What the Edit Fields (Set) Node Does

- Sets, overwrites, or creates new fields in your workflow’s item data.
- Edits field names, values, and data types (string, number, boolean, array, object).
- Allows for manual mapping (using expressions or fixed values) or direct JSON authoring for complex structures.
- Lets you clean, format, and standardize input so every downstream node knows exactly where to find its required data.

Key Use Cases

- Data normalization: Changing data types or fixing values from inconsistent sources (e.g., converting timestamps, splitting/combining fields).
- Creating new variables for logic, API calls, or reporting.
- Restructuring JSON payloads for outgoing requests or database inserts.
- Selectively keeping, removing, or merging certain fields to only output the relevant variables needed downstream.
- Storing static or calculated values (e.g. “status=approved”, “price=total * 0.8”) for use later in the workflow.

How to Set Up and Use the Edit Fields Node

1. Insert the Node: Add after any node generating raw, unstructured, or messy data.

2. Choose Mode:
    - *Manual Mapping*: Drag and drop fields from input, set fixed values or expressions.
    - *JSON Output*: Directly edit or paste a JSON object to transform the entire item.
3. Setting Fields:
    - Specify new field names and values.
    - Change data types, create arrays/objects, and apply formulas with expressions.
    - "Include Other Input Fields" can keep all original data, or you can choose "Keep Only Set Fields" to discard everything but explicitly added fields.
    - Dot notation lets you nest fields (e.g., “user.name” or “address.street”), or you can disable for flat objects.
4. Output: Downstream nodes always reference the cleaned and structured values from your Edit Fields node, making automations reliable and modular.

Best Practices and Tips

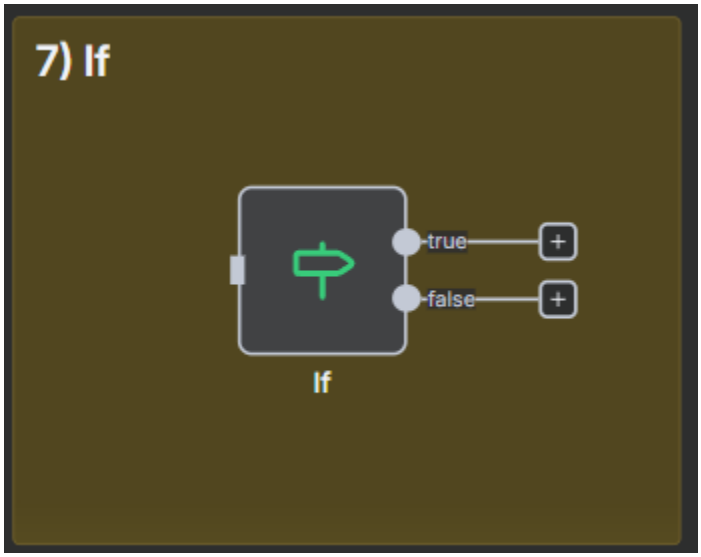
- Use early in your workflow (often right after the trigger or first data node) to standardize all data and prevent issues later on.
- Group logic-related variables into a single object for easier reference and maintenance.
- Combine with Aggregate/Split Out to process data in bulk or individually, then shape it for summary, messaging, or storage.
- Double-check “Keep Only Set Fields” if you need ultra-clean payloads for APIs or databases.

Example Scenario

- An incoming webhook returns {username: "john", timestamp: "2024-01-01T08:00:00Z"}.
- You use the Edit Fields node to:
    - Rename “username” to “user_id”
    - Convert “timestamp” to a formatted date
    - Add a new field “status: active”
- Your downstream nodes only see well-named, correctly typed fields, perfect for integration or reporting.

The Edit Fields node is essential for building robust, readable, and maintainable automations. It gives you total control over your workflow’s data, defining a single point where “truth” and structure are guaranteed.

7. If



The If node in n8n brings decision-making power to your workflows, letting automations run checks and branch out based on your conditions, whether it's equality, comparison, or custom logic.

What the If Node Does

- Evaluates a logical condition against your data (e.g., "status = active", "amount > 1000", "email includes @company.com").
- Sends workflow execution down two branches: "true" (if the condition is met), or "false" (if not).
- Used for advanced filtering, error handling, business logic, and routing different actions based on inputs.

Key Use Cases

- Filtering: Only continue with items meeting a specific criterion, like "score > 80" or "order_status = shipped".
- Quality Gates: Validate incoming records, only process those passing checks.
- Conditional Processing: Route data to different integrations or notifications depending on status/values.
- Error Management: Branch out failed transactions to retry logic or alerting systems.
- Business Logic: Handle premium vs. basic customers, high-priority alerts, or special cases with separate paths.

How to Set Up and Use the If Node

1. Insert the Node: Add after any node where logic-based branching is required (e.g., Set, HTTP Request).
2. Configure Conditions:
    - Choose “Field to Evaluate,” then set “Comparison Operation” (equal to, not equal, greater than, contains, starts with, ends with, etc.).
    - For advanced cases, use expressions, JavaScript-based logic to evaluate complex conditions (e.g., “score > 90 && status == ‘active’”).
3. True/False Branches:
    - Outputs on the “true” connector if the condition matches, “false” connector otherwise.
    - Add different nodes/actions to each branch, automations adapt to any data input.
4. Test Execution: Use test data in development to confirm branching behavior before activating your workflow.

Best Practices and Tips

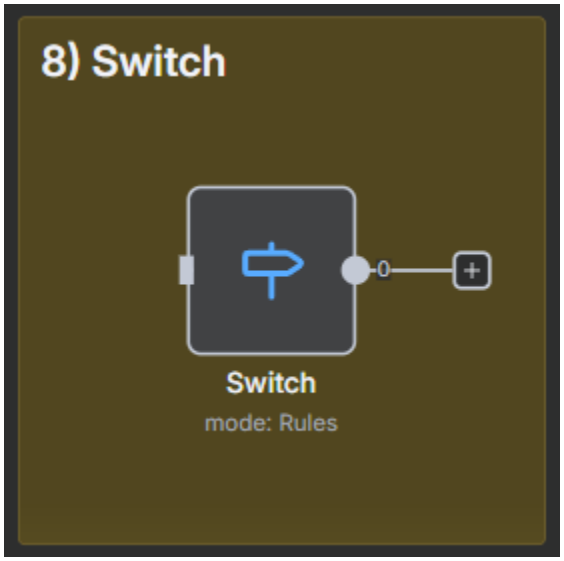
- Break up complex logic using multiple If nodes or combine with Switch nodes for multi-way branching.
- Group related checks into logical blocks for easier debugging and maintenance.
- For checking the existence of fields, use object checks like “{{ $json.items.length > 0 }}”.
- Use in loops to end iterations under certain conditions (e.g., when “nextPage” is false).

Example Patterns

- Basic equality: `{{ $json.status === "active" }}`
- Numeric comparison: `{{ $json.amount > 1000 }}`
- String includes: `{{ $json.email.includes("@company.com") }}`
- Array/object tests: `{{ $json.items.length > 0 }}`

The If node is core to intelligent workflow automation, turning n8n from a static system into a responsive platform that can adapt, filter, and handle business rules dynamically.

8. Switch



The Switch node in n8n is like a supercharged IF node, enabling your workflows to branch off into many different paths, not just two, based on custom routing rules you define.

What the Switch Node Does

- Routes incoming items down one of several output branches, depending on the value of a field or the result of an expression.
- Replaces long chains of IF nodes or complex logic with a single node for clarity and maintainability.
- Supports as many output pads as you need, each with its own matching rule (e.g., 10 possible status codes, 5 cities, or any categories you wish).

Key Use Cases

- Multi-way routing: Send invoices to different email addresses/email nodes based on department ID.
- Action switching: Trigger different actions for each possible event or user role.
- Data categorization: Automatically categorize incoming requests, messages, or tasks by type or urgency.
- Simplifying logic: Replace multiple IF nodes connected in sequence with a single Switch for much neater workflows.

How to Set Up and Use the Switch Node

1. Insert the Node: Place it at any point where your workflow needs to branch by more than two possibilities.
2. Configure Rules: For each output, set a comparison, can be a value check, a match to a string, number, or an expression.
    - Example: Output 1 might match “status = ‘pending’”, Output 2 “status = ‘approved’”, etc..
    - You can add as many outputs (“routes”) as needed and give each a custom label for clarity.
3. Fallback Handling: Optionally set a default route for “no match”, useful for logging or error handling.
4. Downstream Actions: Connect a different processing pipeline, notification, or external call under each branch to handle routed data appropriately.

Best Practices and Tips

- Use Switch nodes to keep branching logic readable and visual, especially when there are several conditions to match.
- Clearly label each output for easier workflow maintenance and debugging.
- For complex conditions (like “amount > 1000 and status = ‘pending’”), use expressions in match rules.
- Combine with IF nodes for hierarchical or nested logic, but let Switch handle all multi-way cases to prevent logic sprawl.

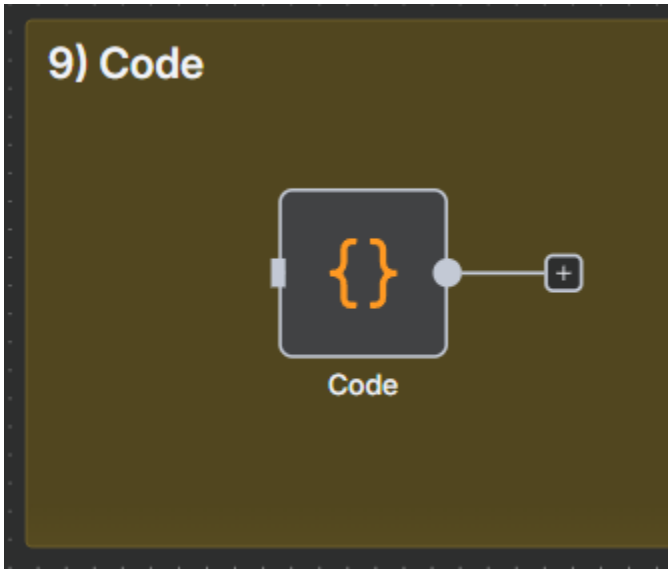
Example Scenario

A webhook submits orders with a field “order_type.”

- The Switch node outputs down different branches for:
    - “retail”
    - “wholesale”
    - “international”
    - “fallback” (if order_type is anything else)
- Each branch then sends data to its own fulfillment or reporting automation chain.

The Switch node is essential for expressive, modular automation in n8n, letting you route, sort, and process data across a wide range of workflow scenarios, all from one node.

9. Code



The Code node in n8n is your go-to tool for manipulating, transforming, and structuring data using custom JavaScript, offering a faster, more predictable, and tightly controlled solution than AI-based or prebuilt nodes for complex operations.

What the Code Node Does

- Executes custom JavaScript on incoming JSON data, either on each item individually or on the whole batch at once.
- Allows you to reshape outputs, add new fields, perform calculations, filter results, or completely restructure your data to fit downstream node requirements.
- Essential for tasks where built-in nodes or expressions aren't enough, such as advanced math, unique JSON reshaping, or dealing with nuanced API response quirks.

Key Use Cases

- Format transformation: Convert API responses (flatten objects, split strings, group fields).
- Custom calculations: Aggregate sums, percentages, or create new computed fields.
- Content manipulation: Extract parts of a string, clean up text, parse dates, reformat emails.
- Data filtering: Keep only items that match complex, multi-condition rules.

- Preparing payloads for APIs, databases, or messaging tools that need precise structure.

How to Use the Code Node Effectively

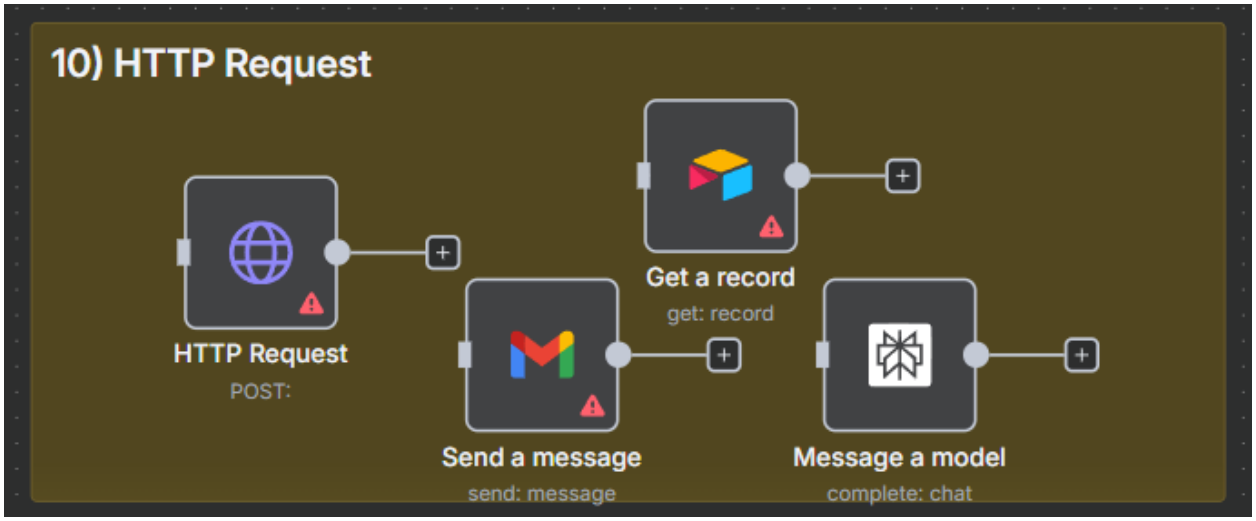
1. Understand Incoming Data: Always inspect the incoming JSON to see what your code will receive. Use the "Execution Data" pane or console.log (for self-hosted setups) to debug.
2. Write Focused Code: Most code works on the variable item.json, representing one workflow item at a time. You can loop through all items or process each individually depending on your needs.
3. Expected Output: Code node must return an array of objects like `[ { json: { ... } }, ... ]`. Returning just an object or array will cause errors, be sure to wrap your outputs properly.
4. Debugging: When errors occur, check that:
    - You didn't accidentally overwrite or remove the json property.
    - Your final output matches what n8n expects (an array of objects).
    - If in doubt, return a simple unchanged item to isolate issues.
5. Let AI Help: For speed, clarity, and accuracy, use AI as a code-writing assistant, provide the error/context and your goal, and let it suggest and fix your code.
6. Edit and Test Iteratively: After writing code, run test executions. If something breaks, share the input, the code, and the error message when seeking help, this clarity speeds up debugging and AI support.

Best Practices and Troubleshooting Tips

- If your code node isn't working, double-check that the structure of your returned data matches exactly what n8n expects ("json" key must point to an object).
- Don't use global imports (like import or export), the code node runs in a sandbox. Use require if self-hosting and external modules are enabled.
- Use n8n's built-in variables, like $input, to reference data from other nodes for clean, reliable code.
- Start simple and build up complexity, test with hardcoded data if needed before plugging in dynamic input.
- For debugging: share error output, the actual code, and a sample of incoming JSON with AI or the n8n community for fast diagnosis.

The Code node is the ultimate tool for bespoke data handling in n8n, powerful, fast, and predictable when you know your data and master the return structure.

10. HTTP Request



The HTTP Request node in n8n is one of the most powerful and versatile tools in your workflow arsenal, enabling you to connect with any API, even when n8n doesn't offer a native integration.

What the HTTP Request Node Does

- Makes outgoing HTTP calls (GET, POST, PUT, DELETE, PATCH, etc.) to any endpoint, letting you pull or push data to third-party services, trigger actions, or integrate legacy or custom APIs.
- Handles everything from standard REST APIs to advanced use cases like authentication, multi-part uploads (for files), and parsing JSON, XML, or plain text responses.
- Can be used to build nearly any automation, fetch weather, submit forms, query databases, control smart devices, invoke cloud functions, and more.

Key Use Cases

- Connecting with platforms not natively supported by n8n, including internal business tools or obscure SaaS products.
- Customizing or extending built-in actions (even for services like Gmail or Airtable) when you need more control or access than the standard node offers.
- Handling API webhooks, interacting with AI services, scraping data, or sending notifications to Slack, Discord, Telegram, etc..
- Building scaleable automations where batching, parallel requests, or rate-limited jobs are required.

How to Use the HTTP Request Node

1. Add the Node: Insert the HTTP Request node where your workflow needs to interface with an outside system.
2. Choose Method and URL: Select the HTTP method (GET, POST, etc.) and specify the full API endpoint you want to hit.
3. Set Authentication:
    - For natively supported auth types, use Predefined Credential Type (OAuth, API key, Basic, etc.).
    - For unsupported/custom APIs, configure headers manually in the node using expression fields or stored credentials.
4. Add Query or Body Parameters:
    - Fill out parameters as key/value pairs or in JSON format, referencing previous node data with n8n's expression system (e.g. {{$json.email}}).
5. Handle Response:
    - The node outputs the API's JSON (or text/XML) as new data items for downstream processing.
    - Use Set, Code, or Edit Fields nodes as needed to transform the results for further steps.
6. Import from cURL:
    - You can paste a cURL command and n8n will auto-fill all relevant node fields, saving massive setup time for complex endpoints.
7. Batching and Loops:
    - Be aware that passing multiple items into the HTTP Request node sends them in parallel batches, manage API rate limits accordingly by controlling batch size or using Split In Batches/Loop Over Items nodes for sequential requests.

Best Practices and Tips

- Use n8n's Credential system, never hardcode keys in node fields.
- Log request/response data for easy debugging.
- Read error responses and manage retries for intermittent failures, especially with rate-limited or flaky APIs.
- Understand the difference between parallel (default) versus sequential requests, sequential is critical for some APIs to avoid bans or throttling.
- When troubleshooting, inspect full API docs, node execution logs, and start with simple GET requests before moving to complex POSTs or auth flows.

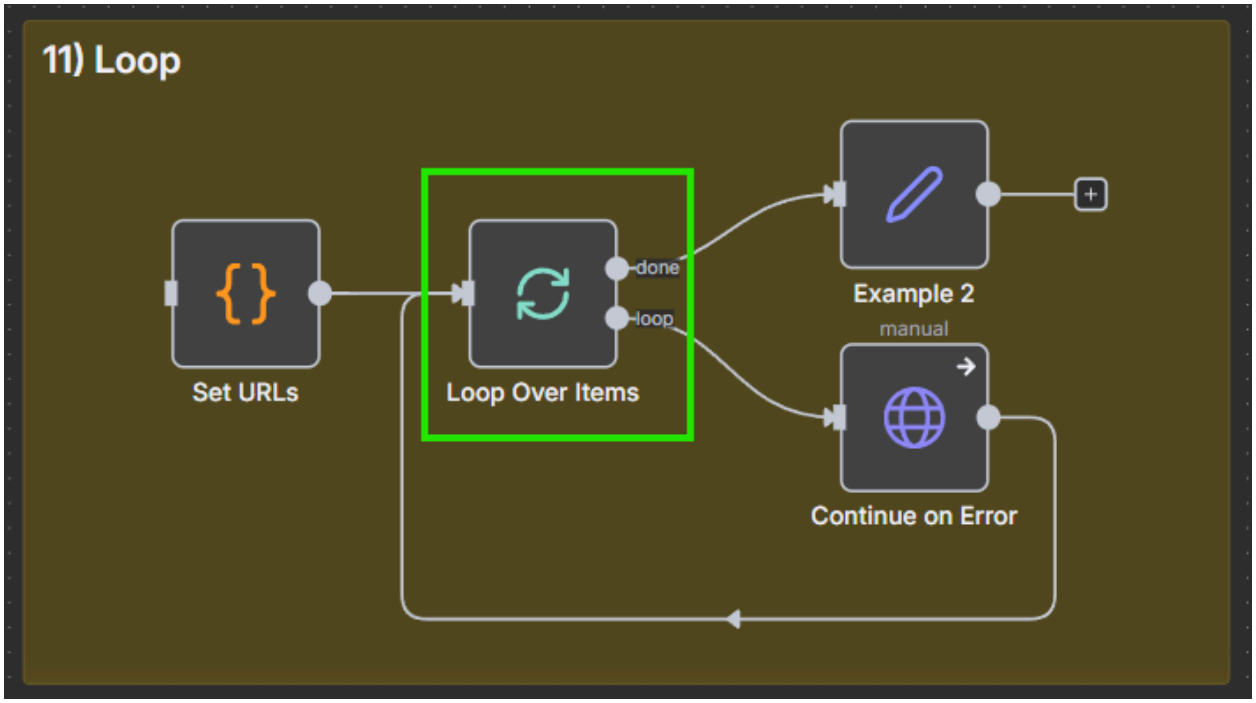
Example Scenario

You need to add data to a CRM system not natively supported by n8n:

- Set the URL to the CRM's REST endpoint.
- Use POST, set proper headers and body fields with expressions.
- Authenticate with an API key using the credential system.
- Handle the JSON response to log success, errors, or further process results downstream.

The HTTP Request node is a must-know resource for every n8n user, unlocking nearly unlimited workflow integrations and pushing the boundaries of automation beyond native nodes.

11. Loop



The Loop Over Items node (sometimes shown as Split in Batches) in n8n is essential for sequential processing, especially when working with large lists and you want to avoid overloading APIs, hitting rate limits, or maxing out memory by processing too many items in parallel.

What the Loop Node Does

- Breaks a list of input items into smaller, manageable batches, often one at a time, processing each sequentially, rather than in bulk or parallel.
- Keeps your automations reliable and resource-efficient, handling API limits, delay requirements, and large-scale data automation.

Key Use Cases

- API Rate-Limiting: Safely send requests to rate-limited APIs, processing just one (or a handful) at a time.
- Large Datasets: Automate tasks like sending emails, updating CRM records, or scraping websites for hundreds/thousands of items without timeouts or system crashes.
- Paginated Workflows: Loop over pages of data, fetch and process each in turn, and optionally handle dynamic item or batch sizing.

- Workflow Throttling: Add pauses (with the Wait node) between batches, ensuring compliance with 3rd-party or backend processing limits.

How the Loop Over Items Node Works

1. Add Node After Data Source: Place the node after a list-producing node (like an API, Code, Google Sheets, or Aggregate node).
2. Set Batch Size: Define the batch size, 1 for strict rate limits, or higher for grouped processing.
3. Connect Your Processing Logic: Downstream, add nodes for your action (e.g. HTTP Request, Email, Data Transformation).
4. Loop Back: Connect the action node back to the Loop node's "loop" input for repeated executions until all batches are processed.
5. Finalize and Exit: When all items are complete, workflow execution moves forward from the Loop node's "done" output.
6. Optional Delays: Insert Wait nodes to throttle request speed between iterations if needed.

Best Practices and Tips

- Prefer Loop Over Items/Split in Batches for API calls to avoid bans or failures with external services.
- Many n8n nodes natively iterate through input arrays, but explicit loops provide precise control and error handling.
- Always set a clear exit condition or finite item list to avoid endless loops, especially when scraping or paginating.
- If you need to process batches in parallel, adjust batch size or combine parallel and loop nodes intentionally.

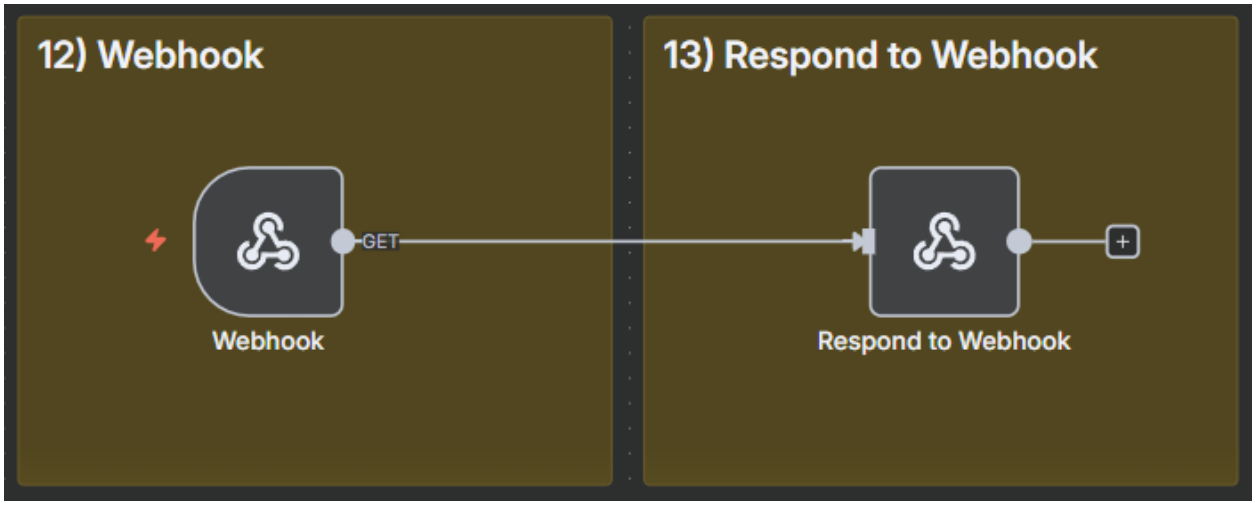
Example Scenario

You have 500 contacts to email (with an API limit of 10 per minute):

- Use Loop Over Items (batch size: 1) and a Wait node (6 seconds).
- The workflow sends one email, pauses, loops for the next, and repeats until all 500 are done, avoiding bans and ensuring stable operation.

The Loop Over Items node makes complex, large-scale automations stable, controlled, and compliant with rate and memory constraints, an essential tool for any serious n8n builder.

12. & 13. Webhook & Respond to Webhook



The Webhook node and Respond to Webhook node together let you build fully custom API endpoints in n8n, empowering you to receive data or events from any external app, even if there's no native integration.

What the Webhook Node Does

- Listens for incoming HTTP requests (like POST, GET, etc.) at a custom URL generated by n8n, instantly triggering your workflow when data arrives from another app, service, or script.
- Makes n8n act as a universal API receiver, any app that can send webhooks or HTTP requests can now trigger your workflow logic.
- Supports key options:
    - Custom paths and HTTP methods.
    - Authentication (optional, for security).
    - Test/Production URLs, use test URLs while building, production URLs when your workflow is live.

What the Respond to Webhook Node Does

- Lets your workflow send a custom HTTP response back to the source that triggered the webhook.
- Use cases: returning calculation results, validation statuses, files, JSON payloads, redirects, or simple text to the original caller, enabling API-like interactions, not just one-way triggers.
- The response can include:
    - All input data

- Just the first item
- A custom JSON or text output
- Redirects and custom response codes, headers, and even JWT tokens for advanced usage.

Key Use Cases

- Instant workflow triggers: React to form submissions, CRM/web app events, payment system notifications, or IoT data.
- Building API endpoints: Create your own lightweight microservices or automation APIs within n8n, where clients both send data and receive structured responses, even building multi-step, multi-node APIs.
- Integration workarounds: When an app or service can't use native triggers, just connect it via webhooks.

How to Use Webhook + Respond to Webhook

1. Add a Webhook Node: Set the HTTP method and custom path; copy the auto-generated URL.
2. Configure the Sender: Paste the webhook URL into the other application's webhook/event settings, or test with Postman/cURL.
3. Set Response Mode: If you want to send a custom response, set the Webhook node’s response mode to "Using 'Respond to Webhook' node."
4. Add Respond to Webhook: Downstream, after processing your data, add a Respond to Webhook node. Configure it to send the response you want (JSON, text, redirect, etc.).
5. Test and Iterate: Use the test URL for active debugging, then shift to the production URL when live.

Best Practices and Tips

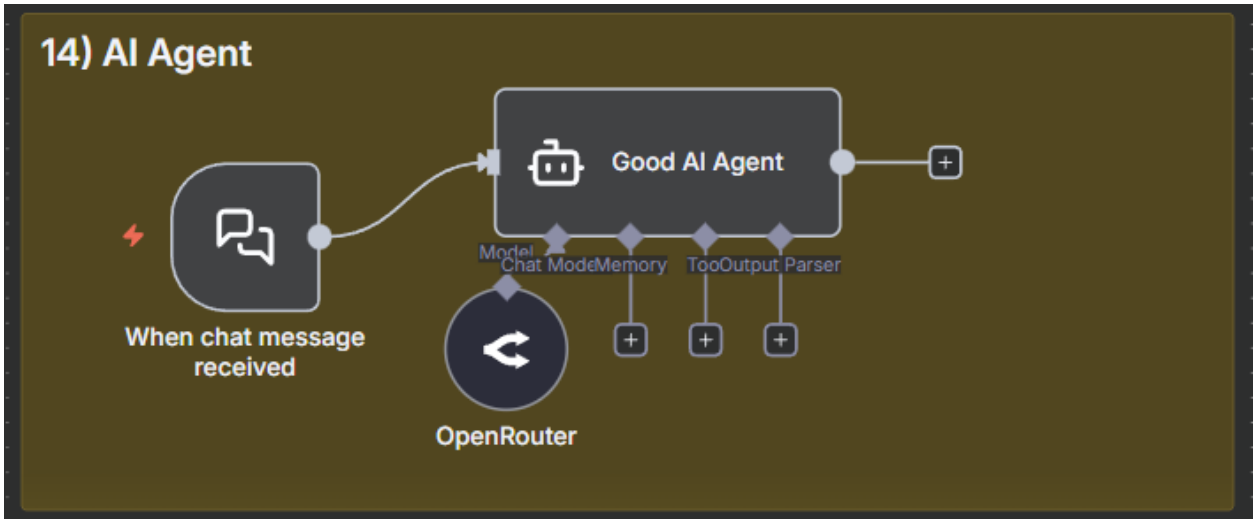
- Always test webhooks with the test URL first; switch to production only for live workflows.
- Be careful with authentication and sensitive data, add auth if exposing endpoints publicly.
- Combine with IF, Switch, and Set nodes for powerful, dynamic API flows.
- Respond to Webhook only returns the first item by default; for collections, ensure your data format matches what the sender expects.

Example Scenario

- You want to accept data from a custom HTML form:
    - Add a Webhook node (POST, custom path), expose the test URL.
    - Connect “Respond to Webhook” node with “Thank you” message or post-processed JSON result downstream.
    - The workflow triggers and replies to the original browser instantly.

The Webhook node and Respond to Webhook node together are essential for making n8n a true integration hub, providing real-time, two-way, event-driven automations beyond standard triggers and actions.

14. AI Agent (LLM)



The AI Agent node in n8n is one of the most powerful tools for workflow automation, letting you integrate advanced chat models (like OpenAI, Gemini, Anthropic, etc.), leverage memory, and chain together dynamic tool usage, all inside your visual workflows.

What the AI Agent Node Does

- Serves as an autonomous reasoning engine, connecting to chat models that can receive prompts, remember prior context, and invoke external “tools” (other nodes) to gather data, make decisions, or automate multi-step processes.
- Empowers complex, context-aware workflows, your agent can answer questions, transform data, fetch info from APIs, update databases, and more, all guided by your instructions and prompt engineering.
- Supports modular tool use: connect tool sub-nodes to give the agent abilities like database queries, web lookups, or even invoking sub-workflows.

Key Use Cases

- Conversational Assistants: Build chatbots capable of maintaining context, answering questions, or executing sophisticated logic over many steps.
- Automated Data Processing: Allow an agent to transform, analyze, summarize, or generate insights from business data, CRMs, support tickets, and more.
- Multi-step Automation: Let the agent decide which actions to take and in what order, fetching, modifying, and synthesizing information across your tech stack.

- Supercharging Integration: Combine chat models with memory and external tools, like APIs, files, or databases, for AI-driven automations you couldn't build with rules alone.

How to Use the AI Agent Node Effectively

1. Add the Node: Insert an AI Agent node into your workflow; attach a chat model (e.g., OpenAI, Gemini, Claude) as the model for reasoning.
2. Connect Tools: Add "tool" sub-nodes, these expose extra skills or APIs (like HTTP requests, database lookups, Slack/Notion actions, etc.) that your agent can call mid-conversation to solve user queries.
3. Prompt Engineering:
    - Craft clear, specific instructions ("system messages") and templates for your desired outputs.
    - Use structured sections (role, instructions, tools, format) to guide and constrain agent responses, ensuring accuracy and relevance.
    - Iterate, test various prompt styles, document what works, and adjust for production reliability.
4. Add Memory: Configure memory settings (vector store, workflow memory, or conversation history) so your agents can recall prior context and act coherently across steps.
5. Configure Advanced Options:
    - Set output formats, fallback models, rate limits, or binary passthrough for complex scenarios.
    - Return intermediate steps if transparency or debugging is important.
6. Trigger and Route: Combine with Webhook or chat trigger nodes for conversational UIs or with API endpoints for programmatic agents.

Best Practices and Tips

- Always define the purpose, role, and response structure in your system message for repeatable, high-quality results.
- Use tools to handle any logic or processing that's unreliable or expensive via LLM, fetch data, handle calculations, or store outputs with sub-nodes.
- Tune memory to the use case: stateless for one-off tasks, conversational for support/chat, or global for persistent reasoning.
- Test with real-world input and iterate on prompt structure and tool composition for production-grade agent performance.
- AI is strongest when it can "think" but use deterministic tools for web scraping, calculations, or integrations.

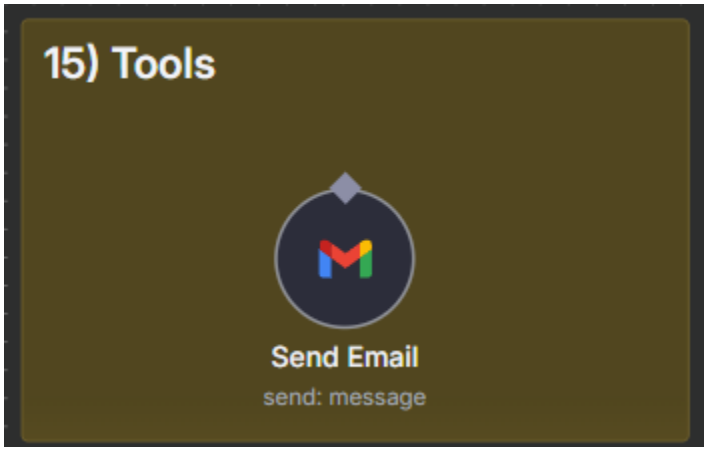
Example Scenario

You want to build a virtual assistant that helps manage YouTube stats, responds to queries, and can email summaries:

- Add an AI Agent node, connect the OpenAI Chat Model, and attach “YouTube Stats” and “Gmail Send” as tool sub-nodes.
- Craft a prompt specifying the agent’s purpose, how to use tools, and the preferred output structure.
- Attach memory so questions like “What were my stats last week?” are answered accurately by referencing previous outputs.

The AI Agent node truly “supercharges” workflows, blending LLM reasoning, memory, and dynamic tool use into a single, flexible automation engine in n8n.

15. Tools



Agent Tools nodes in n8n act as the “hands” of your AI Agent, letting it perform actions on your behalf by calling out to other workflows, APIs, apps, and utilities, amplifying the agent’s power beyond text reasoning.

What Agent Tools Nodes Do

- Expose specific actions (or “tools”) to your AI Agent, such as sending emails, making HTTP requests, querying databases, creating documents/images, or even triggering other n8n workflows as modular sub-processes.
- Each tool node defines an action the agent can invoke, with customizable parameter mapping to ensure dynamic and context-aware operation.
- The AI Agent can decide which tool to use, fill in the right parameters based on a user’s request, and even chain together multiple tools in a single workflow step.

Key Use Cases

- Multi-Action Automation: An agent can answer questions, fetch info, send notifications, create content, and update records, all using the most appropriate tool for the request.
- Dynamic Decision-Making: The agent evaluates incoming tasks and picks the best tool, like deciding to email, call an API, or log a support ticket, based on prompt logic and available options.
- Composable Routines: Tools can include invoking other workflows, enabling powerful composability for modular automation (e.g., chain document processing, email follow-up, and reporting in one “toolbox”).

- AI Content and Media: Let your agent generate images, videos, or documents, retrieve analytics, or handle advanced research, all by chaining model-based agents with no-code tool actions.

How to Use Agent Tools Effectively

1. Add Tools to Your Agent: Connect one or more Tool nodes to your AI Agent node. Each Tool node wraps an action, like sending an email, running a search, fetching API data, or triggering a workflow.
2. Parameter Mapping: In each Tool node, set up dynamic parameters using inputs from the agent's reasoning or the original user request. You can use n8n's expression editor to map required values (recipient, message, search query, file, etc.).
3. Describe Tools in the Prompt: Include detailed tool descriptions, capabilities, and preferred usage in your Agent's system message or prompt. This guidance helps the AI understand when and how to use each available tool.
4. Chaining and Routing: The agent can autonomously call one or more tools as needed, either sequentially (multi-step routines) or conditionally (pick the right tool per use-case).
5. Combining with Sub-Workflows: Use a Tool node to call an n8n sub-workflow, embedding even more complex logic or integrations directly in your agent's toolbox.

Best Practices and Tips

- Clearly name and document each Tool node so the agent chooses logically.
- Provide tool descriptions in the system prompt with when/why/how to use each.
- Use parameter validation to avoid sending incomplete actions (especially when dealing with emails, files, or database records).
- Test by prompting the agent with various requests and checking if it chooses the appropriate tool/action and fills parameters correctly.
- Limit tool permissions and add safeguards when using agents for sensitive or high-impact actions.

Example Scenario

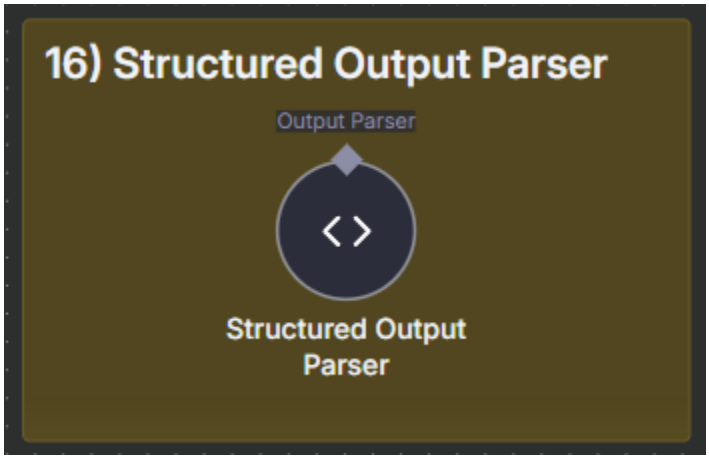
Your AI Agent has four Tool nodes:

- Send Email Tool (SMTP or Gmail node as an action)
- Research Tool (HTTP Request or Web Search)
- Document Creation Tool (Google Docs or Notion node)
- Trigger Report Workflow Tool (calls a reporting sub-workflow in n8n)

When prompted, the agent can read the user's intent ("Email this report to the team") and invoke the right Tool, providing dynamic inputs. If the user asks for "latest competitor stats," it triggers the research tool; if needing to "update project doc," it uses the document tool, automatically, based on context and training.

Agent Tools transform n8n AI from a reasoning engine into a full-powered automation assistant, capable of making decisions and taking actions across your software stack.

16. Structured Output Parser



The Structured Output Parser node in n8n lets you force LLM (Large Language Model) or AI Agent nodes to produce output in a precise JSON structure, empowering you to reliably extract multiple objects, arrays, strings, or deeply nested data types with total control over the output format.

What the Structured Output Parser Node Does

- Validates and parses the raw output from an LLM or AI Agent according to a defined JSON schema, ensuring responses adhere to your required structure every time.
- Supports both manual schema definition and schema generation from a JSON example, making it easy to get up and running with complex, nested data.
- Reduces downstream error handling and post-processing, so you can immediately consume, transform, or pass structured data to other nodes like Set, Aggregate, or HTTP Request.

Key Use Cases

- API/Automation Consistency: Build AI tasks that always output the same shapes, no guessing or “extract-by-string” hacks.
- Multi-field Responses: Extract lists, objects, scoring arrays, FAQs, or hybrid types (like an array of objects each with their own properties) directly from a single AI step.
- Data Integrity: Ensure LLM outputs match the requirements of subsequent automations, such as sending to a database, spreadsheet, or API endpoint without extra parsing or error-prone transforms.

How to Use the Structured Output Parser Node

1. Enable in AI Node: In the AI Agent or LLM root node, enable “Require Specific Output Format.” This reveals an output parser attachment point.
2. Attach the Parser: Add the Structured Output Parser node; it now controls the output for this workflow branch.
3. Define Schema:
    - Use “Generate from JSON Example” for quick setup, just input a sample of your desired output, and n8n builds the schema.
    - For advanced uses, “Define using JSON Schema” lets you specify any level of nesting, required fields, and data types manually.
4. Map/Format Output: Downstream, you can confidently map or consume fields due to predictable output (e.g., {{ $json.invoices.total }} or {{ $json.summary }}).
5. Error Handling: If the output doesn’t match your schema, the node will throw an error, highlighting prompt/model alignment issues early.

Best Practices and Tips

- Always include explicit example outputs or schema requirements in your LLM/agent prompts to reinforce adherence by the AI model.
- Use manual JSON Schema for tighter control and validation, especially if other systems rely on strict data types or required fields.
- For agents, be aware that intermediate tool calls may not always respect the output specification, best to parse final output only, not intermediary steps.
- Use follow-up LLM chains or post-processing Function nodes to “auto-fix” or repair occasional malformed outputs with unreliable models.

Example Scenario

Suppose you ask the AI Agent “Summarize sales by region, and list top 3 customers for each region.”

- Provide an example output or schema like:
- json

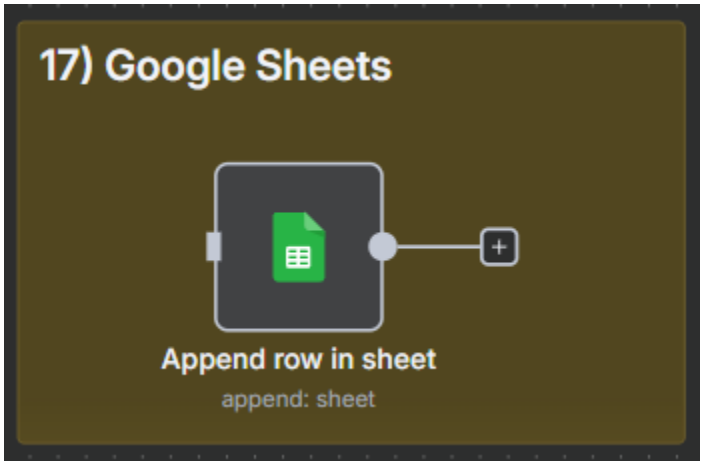
```
{
  "regions": [
    {
      "name": "North",
      "total": 123456,
      "top_customers": ["Acme", "Beta", "Gamma"]
    }
```

```
  ]
}
```

-
- The Structured Output Parser will ensure the agent delivers exactly this form, ready for analysis or reporting.

The Structured Output Parser node is a core tool for anyone building reliable, production-grade automations and AI-powered flows in n8n: it eliminates unpredictable text parsing and delivers rock-solid data structures in every run.

17. Google Sheets



The Google Sheets node in n8n is a cornerstone for workflow automation, serving as both a flexible, cloud-based temporary database and a live reporting engine you can read from, write to, and sync across different systems and automations.

What the Google Sheets Node Does

- Allows your n8n workflows to interact with any Google Sheet for reading, writing, appending, updating, and deleting rows, serving as a semi-structured database you control.
- Lets you keep workflow state, share data across different automations, and collaborate with team members in a familiar spreadsheet interface.
- Can be paired with triggers (like “Row Added/Updated”) or actions (like “Append Row,” “Update Row,” “Read Sheet”) for total data sync and automation power.

Key Use Cases

- Temporary or Persistent Storage: Log lead data, user actions, form submissions, or transient automation states, retrievable later by any workflow.
- Status & Tracking: Use columns for “status,” “error,” “needs follow-up,” etc., so automations can check if/when each row needs to be triggered or updated.
- Data Sync: Regularly back up app data (e.g., from Airtable or APIs) or sync to other platforms in real-time, providing both reporting and easy troubleshooting.
- Continuous Updates: Add new rows with “Append,” update existing ones (e.g., when statuses change), or remove obsolete entries, ensuring your spreadsheet mirrors the latest workflow state.

How to Use Google Sheets Nodes Effectively

1. Authentication: Connect your Google account with n8n using OAuth credentials. You may use built-in credentials for most cases or set up your own via the Google Cloud Console for custom security.
2. Configure the Node:
    - Choose the spreadsheet and worksheet/tab.
    - Pick your operation: Append, Update, Read, Delete, Lookup, etc..
    - For updates/deletes, specify the key (unique column) to identify which row/record to update.
    - Map fields using n8n expressions (e.g., {{ $json.email }}) for dynamic data population.
3. Advanced Tricks:
    - Use Lookup or Filter operations in n8n to find the exact row you want, then update it conditionally.
    - Chain multiple Google Sheets nodes for complex logic (sequential updates, aggregated reporting, etc.).
    - Sync status columns or flags to coordinate multi-stage automations or resume/retry as needed.

Best Practices and Tips

- Always use a unique key (like “ID” or “email”) for reliable updates, Google Sheets nodes identify rows by your chosen key column.
- For large sheets (10k+ rows), consider intermediate lookups and paging to avoid slowing down workflows.
- Set up error handling and retries for occasional API timeouts or Google-side issues.
- For critical or confidential data, beware of spreadsheet access/visibility, ensure correct permissions in Google Drive.

Example Scenario

Suppose you want to log every new customer sign-up:

- Start with a webhook/form trigger.
- Use the “Append Row” operation in Google Sheets node to add new entries.
- Periodically, run a scheduled workflow that reads the sheet, checks for “status = pending,” and triggers follow-up automations for those rows.
- As automations progress, use “Update Row” to change their statuses to “processed,” “error,” or “complete,” keeping everything visible and audit-ready in the sheet.

Mastering Google Sheets with n8n unlocks vast automation potential, turning a humble spreadsheet into a real-time dashboard, an automation queue, or an always-available mini-database for any workflow.
