Lead Gen Pack — Setup Guide

Import steps • Configuration • curl tests • FAQ 3 n8n workflows for lead capture and enrichment

Spis treści

• Lead Gen Pack — Setup Guide

Lead Gen Pack — Setup Guide

Lead Gen Pack - Setup Guide

Three importable n8n workflows for lead capture and enrichment. Everything runs inside n8n; the only external services are an OpenAI-compatible chat completions endpoint and an SMTP server.

Contents

`w1-lead-capture-enrich.json`	**Lead capture and AI qualification**	**Webhook** `POST /lead`	**Webhook, HTTP Request, IF, Send Email, Set**
`w2-feed-monitor.json`	Feed monitor and email digest	Schedule, every 6 hours	Schedule Trigger, Set (config), HTTP Request, IF, Set (digest), Send Email
`w3-bulk-enrich.json`	Bulk lead enrichment to CSV	Webhook `POST /enrich`	Webhook, Set, Split Out, Split In Batches, HTTP Request, Merge, Spreadsheet File

All three use only core nodes (`n8n-nodes-base.*`). No community nodes are required.

Prerequisites

- n8n 1.60 or newer, self-hosted (Docker) or n8n Cloud. The bundled node versions are: `webhook` 2, `httpRequest` 4.2, `set` 3.4, `if` 2.2, `emailSend` 2.1, `scheduleTrigger` 1.1, `splitInBatches` 3, `spreadsheetFile` 1, `merge` 3. Older 1.x builds may display a version warning on import. [verify] against your installed version.
- An API key for any OpenAI-compatible `/v1/chat/completions` endpoint.
- An SMTP account you can send from (Gmail, Fastmail, Postmark, your own relay, etc.).
- A reachable base URL for webhooks (W1 and W3 only). For local testing, n8n's own host is enough.

Import a workflow

Download the `.json` file you want to import.

Open n8n and go to the workflow editor.

Open the `...` menu at the top right of the editor and choose Import from File.

Some builds expose the same entry as Overview -> Add workflow -> Import from file. [verify]

Pick the `.json` file. The workflow opens as a new, unsaved workflow.

Save it (`Ctrl/Cmd + S`).

Repeat for the other two files. Each import creates a separate workflow.

Importing does not activate anything. Webhooks only answer on the production URL after you activate the workflow, and the schedule trigger only fires when the workflow is active.

Set the API key

W1 and W3 read the key from the environment variable `OPENAI_API_KEY` inside the HTTP Request node:

```
Authorization: =Bearer {{$env.OPENAI_API_KEY}}
```

Environment variables must be present in the n8n process, not in your shell. Restart n8n after adding one, then re-run the workflow - expressions are resolved per execution.

Docker, inline:

```bash
docker run -d --name n8n \
-p 5678:5678 \
-e OPENAI_API_KEY="sk-your-key" \
-v n8n_data:/home/node/.n8n \
docker.n8n.io/n8nio/n8n
```

Docker Compose:

```yaml
services:
n8n:
image: docker.n8n.io/n8nio/n8n
ports:
- "5678:5678"
environment:
- OPENAI_API_KEY=sk-your-key
- N8N_BLOCK_ENV_ACCESS_IN_NODE=false
volumes:
- n8n_data:/home/node/.n8n
volumes:
n8n_data:
```

Notes on the second variable: n8n exposes environment variables to expressions unless env access is blocked. If `N8N_BLOCK_ENV_ACCESS_IN_NODE=true` is set on your instance, `{{$env.OPENAI_API_KEY}}` resolves to an empty string and the OpenAI call returns `401`. Leave it unset or `false`. [verify]

n8n Cloud: environment variables are available on paid plans only. [verify] If your plan does not support them, replace the header value instead:

Create a Header Auth credential: name `Authorization`, value `Bearer sk-your-key`.

Open Qualify Lead (OpenAI HTTP) (W1) and Enrich Batch (OpenAI HTTP) (W3).

Under Authentication choose Generic Credential Type -> Header Auth and select the credential.

Clear the `Authorization` entry in the node's header list so it is not sent twice.

Configure SMTP

Go to Credentials -> Add credential and choose SMTP (the credential type used by *Send Email*).

Fill host, port, user and password, and enable TLS for ports 465 (implicit) or 587 (STARTTLS). Gmail requires an app password with 2-Step Verification enabled and the host `smtp.gmail.com`. [verify]

Save and test the credential.

Open each email node, select your SMTP credential, then fill From Email and To Email:

- W1: Alert Sales (Email SMTP)
- W2: Send Digest (Email SMTP)

The JSON ships with a placeholder credential reference named `Gmail` that does not exist in your instance. Until you select your own credential, the node shows a missing-credential warning; this is expected and does not affect import. W1 and W2 also ship with empty `From Email` and `To Email` values - fill both before the first production run.

Webhook URLs and test calls

Each workflow has two URLs:

- Test: `https://<your-n8n-host>/webhook-test/<path>` - valid after clicking Listen for test event.
- Production: `https://<your-n8n-host>/webhook/<path>` - valid while the workflow is active.

W1 uses path `lead`, W3 uses path `enrich`. Both webhooks respond immediately, so the HTTP call returns before the AI step completes; the work continues in the execution log.

W1 test call:

```bash
curl -X POST "https://n8n.example.com/webhook/lead" \
-H "Content-Type: application/json" \
-d '{
"name": "Ada Lovelace",
"email": "ada@example.com",
"company": "Analytical Engines",
"message": "We need pricing for 40 seats, evaluating this quarter"
}'
```

Expected: an immediate `200` plus a JSON body from the webhook, and an email when the model scores the lead `>= 7`. Lower-score leads take the Archive (No Action) branch and send nothing.

W3 test call:

```bash
curl -X POST "https://n8n.example.com/webhook/enrich" \
-H "Content-Type: application/json" \
-d '{
"leads": [
{ "name": "Ada Lovelace", "email": "ada@example.com", "company": "Analytical Engines" },
{ "name": "Grace Hopper", "email": "grace@example.com", "company": "Navy Systems" }
]
}'
```

A bare array is also accepted:

```bash
curl -X POST "https://n8n.example.com/webhook/enrich" \
-H "Content-Type: application/json" \
-d '[{ "name": "Ada Lovelace", "email": "ada@example.com", "company": "Analytical Engines" }]'
```

The CSV is produced at the end of the execution, not in the HTTP response. Open Executions -> the run -> Convert to CSV (Spreadsheet) and download the binary output (`data`, `enriched-leads.csv`).

Configure each workflow

W1 - Lead capture

Score threshold	Score >= 7 (IF), condition right value `7`
Lead fields accepted	`name`, `email`, `company`, `message` in the webhook body
Prompt	Qualify Lead (OpenAI HTTP), `messages[0].content` inside the JSON body
Model	`gpt-4o-mini` in the JSON body
Alert recipients	Alert Sales (Email SMTP), `From Email` / `To Email`

The model is asked for a single integer. The IF node reads it with `parseInt(...)` and falls back to `0` when the reply contains no leading number, so a chatty reply archives the lead instead of failing the workflow.

W2 - Feed monitor

Feed URL	Config, `feedUrl`
Look-back window	Config, `maxAgeHours` (default `7`)
Schedule	Every 6 Hours, interval `hours` / `6`
Digest recipient	Send Digest (Email SMTP), `From Email` / `To Email`

The feed must return a JSON object with an `items` array at the top level. Recognized per-item keys:

```
title, url (or link), date_published (or pubDate, isoDate, published, created_at)
```

This is the shape used by the JSON Feed spec and by RSS-to-JSON bridges. Raw RSS/XML is not parsed:

the pack deliberately contains no XML node. Point `feedUrl` at a JSON conversion of the feed.

Newness is decided by comparing each item's date against `now - maxAgeHours`. Use a window slightly wider than the schedule interval (6h schedule, 7h window) so consecutive runs overlap instead of leaving gaps. See Known limitations below for the consequences.

W3 - Bulk enrichment

Batch size	**Batch Leads (2),** `Batch Size`
Enrichment prompt	Enrich Batch (OpenAI HTTP), `messages[0].content`
Model	`gpt-4o-mini` in the JSON body
Output file	Convert to CSV (Spreadsheet), `fileName` and `binaryPropertyName`

The prompt asks for a JSON array of objects with `name`, `email`, `company`, `industry`, `size`, `summary`. Output is written with `headerRow: true`.

Customizing the prompt

The prompt is a JavaScript string inside the node's JSON body, so the object keys are quoted and the whole thing is wrapped in `JSON.stringify({...})`. Edit the `content` strings only.

W1 qualification prompt:

```text
You are a B2B lead qualification engine. Score the lead from 0 to 10 based on company fit,
clarity of need and buying intent. Reply with a single integer from 0 to 10 and nothing else.
```

Rules that keep W1 working when you rewrite it:

- Keep the reply contract as one integer. The IF node parses with `parseInt`, so `7`, `7/10` and `Score: 7` all work; `high` does not and is treated as `0`.
- Add your qualification criteria to the same `content` string; do not add new messages unless you also keep the user message that carries the lead fields.
- `max_tokens` is `16`. Raise it only if the reply contract grows.

W3 enrichment prompt - the important part is the output contract:

```text
Reply with a JSON array only and no markdown fences. Keep the input order.
```

Changing the requested keys changes the CSV header row, which is generated from the first item.

Other common edits: swap `model` for `gpt-4o`, `o4-mini` or any model your endpoint exposes; raise `temperature` only if you want variety (deterministic scoring uses `0`); change the endpoint `url` for Azure OpenAI, OpenRouter, Together, vLLM or Ollama-compatible servers - keep the `/v1/chat/completions` request shape and the `Authorization` header your provider expects.

FAQ

1. Do I need a paid OpenAI account?

You need API credits. A ChatGPT subscription does not cover API usage; create a key in the API dashboard and add billing there.

2. Can I use a local or non-OpenAI model?

Yes, if it exposes an OpenAI-compatible chat completions endpoint. Change the URL in the HTTP Request node and adjust the `Authorization` header (some local servers need no auth). Functionality that depends on the model: W1 needs a single-integer reply, W3 needs a bare JSON array.

3. Which model should I put in the JSON?

`gpt-4o-mini` is the default because scoring and one-line enrichment are cheap tasks. Any model that follows the reply contract works; larger models cost more per lead and are not required for this format.

4. Will W2 email me the same item twice?

Yes, for items that stay inside the `maxAgeHours` window across consecutive runs. The workflow is deliberately stateless: it compares item dates to the clock instead of storing a last-seen marker (see Known limitations). To reduce overlap, lower `maxAgeHours` to just above the interval; to remove it, add a de-duplication step.

5. Can W3 return the CSV as an HTTP download?

Not with the node set in this pack. The webhook answers immediately (`responseMode: onReceived`), so the file is written at the end of the execution instead. To return it over HTTP, set the webhook `Response Mode` to `Using Respond to Webhook node`, add a Respond to Webhook node with `Respond With: Binary File` after the CSV node, and reference the `data` property. [verify] on your n8n version.

6. Where does my lead data go?

Lead payloads go to your n8n instance, to the AI endpoint, and (as an email body) to your SMTP provider. n8n also stores execution data, including the payloads, in its own database. Set `EXECUTIONS_DATA_PRUNE=true` and `EXECUTIONS_DATA_MAX_AGE` to limit retention, or disable saving successful executions in the workflow settings. [verify]

Known limitations

- W2 is time-window based, not stateful. "New since last run" is approximated by

`now - maxAgeHours`. A true last-seen marker needs `$getWorkflowStaticData`, which is only available in a Code node; the pack stays inside the core node set without Code.

- W3 receives a JSON array as a single n8n item. n8n's Webhook node puts the whole body in `json.body`; a `Split In Batches` node then loops over items, and an array payload is one item, so the batch runs once and the CSV carries one row per batch. To get one row per lead, insert a Split Out node between Normalize Leads (Set) and Batch Leads (2), set field to `leads`, and keep `Batch Size` at 1 for one lead per API call. The CSV header row follows the first item's keys, so flattening the model reply into columns with a Set node gives a cleaner file.
- W1 scores on a single integer. No structured reasoning is returned to the email; the raw model reply is included so you can see what it said.
- No authentication on the webhooks. Anyone with the URL can post. Add webhook authentication (Basic, Header or JWT) before exposing the URL publicly.
- No retry or error branch. A failed HTTP call fails the execution and is visible in Executions; add node-level *Retry On Fail* or an Error Trigger workflow if you need it.

Troubleshooting

"Credential not found" or a red credential field on an email node

The imported workflow references a credential id that does not exist in your instance. Open the node, select your SMTP credential, then save. Fill `From Email` and `To Email` at the same time.

Import does nothing / "The file is not valid JSON"

The file was edited and lost valid JSON (usually a trailing comma or a straight quote replaced by a typographic quote). Re-import the original file; JSON strings must use `"` and expressions must keep their inner single quotes. There are no comments in these files.

Node shows "unknown node type" or the version warning is red

The instance is older than the node versions listed in Prerequisites. Upgrade n8n and re-import.

Webhook returns 404

Three usual causes: the workflow is not active (production URL), you are calling the production URL while the editor is only in test mode (use `/webhook-test/` and click Listen for test event), or the path was changed in the webhook node.

OpenAI returns 401

`OPENAI_API_KEY` is not visible to the n8n process. Check that the variable is set on the container (not only in your shell), restart n8n, and confirm env access is not blocked. On n8n Cloud without environment variables, use a Header Auth credential instead of `{{$env.OPENAI_API_KEY}}`.

OpenAI returns 404 or 400

The `url` field must point at the full chat completions path, for example `https://api.openai.com/v1/chat/completions`. A base URL without the path returns 404; a provider that expects a different body shape returns 400.

W1 never sends an email

Open the execution and check the IF node output. A score below 7 takes the archive branch by design; `0` means the model reply had no leading integer (see Customizing the prompt).

W1 or W2 email fails with "Invalid email"

`From Email` / `To Email` are empty. Fill both. W2's subject and body come from the Set node, so they are empty only if the execution never reached it.

W2 runs but sends nothing

Check that the feed returns `{"items":[...]}`, that item dates are parseable ISO/RFC dates, and that at least one item is newer than `now - maxAgeHours`. Items without a recognized date key are treated as timestamp `0` and ignored.

W3 produces one CSV row or an empty file

Expected for an array payload - see Known limitations. Empty output usually means the webhook body had no `leads` array and the endpoint received `Leads: []`.

"Expression error: cannot read properties of undefined"

A payload field is missing or the node order changed. The expressions guard missing lead fields but assume the webhook body, the `items` array and the model reply exist.

IF node reports a type validation error

The condition compares numbers. If you replaced the score expression with something that returns text, wrap it in `Number(...)` or restore the original expression.