Client Ops Pack — Setup Guide

Import steps • Configuration • curl tests • FAQ 3 n8n workflows for proposal intake, onboarding and follow-up

Spis treści

• Client Ops Pack — Setup Guide

Client Ops Pack — Setup Guide

Client Ops Pack - Setup Guide

Three importable n8n workflows for a small agency or solo consultant: inbound proposal triage, client onboarding, and a 14-day follow-up sequence. Everything runs inside n8n. The only external services are an OpenAI-compatible chat completions endpoint and an SMTP server you send from. All three workflows use core nodes only (`n8n-nodes-base.*`). No community nodes, no sub-workflows, no shared credentials between them.

What is in the pack

`w1-proposal-intake.json`	**Score inbound proposals, alert on high fit**	**Webhook** `POST /proposal`	**5**	**1 internal alert**
`w2-client-onboarding.json`	Onboarding sequence, day 0 to day 3	Manual trigger (pinned sample)	7	3 to the client
`w3-follow-up-sequence.json`	Follow-up after a sales call, day 0 to day 14	Webhook `POST /start-followup`	8	4 to the contact

Node versions used: `webhook` 2, `httpRequest` 4.2, `set` 3.4, `if` 2.2, `emailSend` 2.1, `manualTrigger` 1, `wait` 1.1.

Prerequisites

- n8n 1.60 or newer, self-hosted or Cloud. Older builds may show a node-version warning on import. The listed versions are the pack target, not a measured lower bound. [verify] against your build.
- An API key for any OpenAI-compatible `/v1/chat/completions` endpoint (W1 only).
- An SMTP account you can send from (W1, W2, W3).
- A publicly reachable hostname for the webhooks (W1, W3). The n8n host itself is enough for local testing. Webhook URLs must be reachable from whatever sends the proposal or starts the follow-up.

Import the workflows

Download the `.json` file you want.

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 files. Each import creates a separate workflow.

Importing does not activate anything.

- W1 and W3 only answer on the production URL (`/webhook/...`) after you activate the workflow. In

the editor, use Listen for test event and the `/webhook-test/...` URL instead.

- W2 has a manual trigger, so it never fires on its own. You run it from the editor (see below).

Both webhooks respond immediately (`responseMode: onReceived`), so the sender gets `200` before the AI call or any email has happened. The rest of the run continues in Executions.

Set the API key

W1 reads the key from the environment variable `OPENAI_API_KEY`:

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

The variable must be present in the n8n process, not only in your shell. Restart n8n after adding it; 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:
```

If `N8N_BLOCK_ENV_ACCESS_IN_NODE=true` is set on your instance, `{{$env.OPENAI_API_KEY}}` resolves to an empty string and the request 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, use a Header Auth credential instead:

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

Open Score Proposal (OpenAI HTTP) in W1.

Set Authentication to Generic Credential Type -> Header Auth and select the credential.

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

The endpoint is `https://api.openai.com/v1/chat/completions`, model `gpt-4o-mini`. For Azure, OpenRouter, Together, vLLM or Ollama, change the URL and the `Authorization` header to whatever that provider expects, keeping the `/v1/chat/completions` request body.

Configure SMTP

Go to Credentials -> Create credential and choose SMTP.

Fill host, port, user and password. Port 465 uses implicit TLS, 587 uses STARTTLS. Gmail needs a host of `smtp.gmail.com` and an app password with 2-Step Verification enabled. [verify]

Save, then run the credential test.

Open every node named ... (Email SMTP) or Send ... (8 nodes in total across the pack) and select the credential in the Credential to connect with field.

The JSON ships with a placeholder credential reference (`id: REPLACE_WITH_SMTP_CREDENTIAL_ID`, `name: SMTP`). n8n shows a missing-credential warning until you select your own credential; this does not affect import. The referenced credential is never auto-created - create it yourself.

`From Email` is empty in all 8 email nodes and must be filled with an address your SMTP account is allowed to send as. `To Email` for client-facing emails is an expression, so it needs no editing:

- W2: `={{$json.email}}` on day 0 and `={{ $('Build Welcome Email').first().json.email }}` later.
- W3: `={{ $json.body.email }}` on day 0 and `={{ $('Start Follow-Up (Webhook)').first().json.body.email }}` later.

The later emails reference the trigger or Set node by name instead of `$json` because a Wait node resumes with the data it received, not necessarily the previous email node's output. Keep those node names if you edit a workflow.

W1 sends an internal alert, not an email to the lead, so its To Email is deliberately empty. Fill it with the inbox that should receive proposal alerts.

W1 - Proposal intake and fit score

Flow: `Proposal Webhook` -> `Score Proposal (OpenAI HTTP)` -> `Score >= 7 (IF)` -> `Alert High-Fit (Email SMTP) (true) or Archive (No Action)` (false).

Webhook URL and test call

- Test: `https://<your-n8n-host>/webhook-test/proposal` after clicking Listen for test event.

- Production: `https://<your-n8n-host>/webhook/proposal` while the workflow is active.

```bash
curl -X POST "https://n8n.example.com/webhook/proposal" \
-H "Content-Type: application/json" \
-d '{
"name": "Ada Lovelace",
"email": "ada@example.com",
"company": "Analytical Engines",
"budget": "15000",
"scope": "Website rebuild plus CRM setup",
"notes": "Wants to start in Q1"
}'
```

Expected response: `200` with `{"message":"Workflow was started"}`. Then, within a few seconds:

- A score of 7 or more sends one alert email containing the score, the one-line summary, every submitted field, the raw model reply and a UTC timestamp.
- A score below 7 sends nothing. The Archive (No Action) node emits `{status: "archived", email, score, archivedAt}` and the run ends. This is intentional: low-fit requests are parked in the execution log instead of in your inbox.

All six body fields are optional. Missing fields render as empty strings rather than failing the expression. `budget` may be a number or a string.

Configuration points

Score threshold	Score >= 7 (IF), condition right value `7`
Alert recipient	Alert High-Fit (Email SMTP), `To Email` (empty on import)
Sender address	same node, `From Email`
Prompt	Score Proposal (OpenAI HTTP), `messages[0].content` in `jsonBody`
Model	`gpt-4o-mini` in `jsonBody`

The AI reply contract

The prompt asks for exactly one line in this format:

```
8|Clear scope, realistic budget, ready to start this quarter.
```

- The score is read with `parseInt`, so the reply must start with the digits. `8`, `8/10` and `8 - strong` all score `8`. A reply that starts with anything else, such as `Score: 8` or `high`, parses as `0` and takes the archive branch.
- The summary is everything after the first `|`. If the model omits the pipe, the alert shows

`n/a`.

- `max_tokens` is `80`. Raise it if you change the summary length.
- The raw model reply is included in the alert, so a malformed reply is visible without reading the execution log.

W2 - Client onboarding

Flow: `Manual Start` -> `Build Welcome Email` (Set) -> `Send Welcome (Day 0)` -> `Wait 24h` -> `Send Check-In (Day 1)` -> `Wait 72h` -> `Send Milestone Ask (Day 3)`.

How to run it

Open W2 and select Manual Start.

Click Execute workflow (bottom of the editor, `Ctrl/Cmd + Enter`).

The run pauses at each Wait node. The emails go out on this schedule from the moment you start:

Send Welcome (Day 0)	none	0
Send Check-In (Day 1)	24 h	1 day
Send Milestone Ask (Day 3)	72 h	4 days

The labels name the sequence step, not the exact wall-clock day. If you need day 3 at exactly 72 hours after day 0, change the second wait to `48`.

The sample client

The workflow ships with one pinned sample item on Manual Start:

```json
{
"name": "Jane Client",
"email": "jane@example.com",
"projectStart": "2026-12-01",
"deliverables": "Website + CRM setup"
}
```

Pinned data is what a manual test run uses. To onboard a real client, either edit the pin before running, or - for repeated use - replace Manual Start with a real trigger (Form Trigger, Webhook or a CRM node) and keep the same four field names. The Build Welcome Email Set node reads `name`, `email`, `projectStart` and `deliverables`, so any trigger that supplies those fields works.
To change the sample: unpin the node in the editor, run once with pasted JSON, or edit `pinData` in the file before import. The pinned values are not used when the workflow is active, because a manual trigger has no production event. [verify] the pin/unpin menu label on your n8n version.

Email content

The welcome subject and body are built in Build Welcome Email and read by the day 0 email, so

there is one place to edit the opening message. The day 1 and day 3 emails are written directly in their nodes and pull the client name, start date and deliverables from `$('Build Welcome Email').first().json`.

All three emails end with:

```
Best regards,
The team
```

Replace that signature with your name or agency in the three email nodes (W2) and the four email nodes (W3).

W3 - Follow-up sequence

Flow: `Start Follow-Up (Webhook)` -> `Send Thanks (Day 0)` -> `Wait 3 Days` -> `Send Value-Add (Day 3) -> Wait 4 Days -> Send Case Study (Day 7) -> Wait 7 Days -> Send Gentle Close (Day 14)`.

Webhook URL and test call

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

```bash
curl -X POST "https://n8n.example.com/webhook/start-followup" \
-H "Content-Type: application/json" \
-d '{"name": "Mark Owner", "email": "mark@example.com"}'
```

Expected response: `200` with `{"message":"Workflow was started"}`. The first email sends immediately; the remaining three send 3, 7 and 14 days after that.

Send Thanks (Day 0)	none	0
Send Value-Add (Day 3)	72 h	3 days
Send Case Study (Day 7)	96 h	7 days
Send Gentle Close (Day 14)	168 h	14 days

The case study email contains a generic story, not your results. Replace it with a real case study before selling the sequence as yours.

Changing wait intervals and email templates

Wait intervals

Every Wait node stores a number and a unit. The pack uses hours so the JSON stays comparable across nodes:

W2	Wait 24h	{"amount": 24, "unit": "hours"}	1 day
W2	Wait 72h	{"amount": 72, "unit": "hours"}	3 days
W3	Wait 3 Days	{"amount": 72, "unit": "hours"}	3 days
W3	Wait 4 Days	{"amount": 96, "unit": "hours"}	4 days
W3	Wait 7 Days	{"amount": 168, "unit": "hours"}	7 days

In the editor, open the node and change Wait Amount / Wait Unit (`seconds`, `minutes`, `hours`, `days`). Keep the node names if you edit the JSON, because later email nodes reference upstream nodes by name.

Lower values are convenient for testing: set all three W3 waits to `60` seconds, run the sequence, and you receive all four emails within about three minutes.

Email templates

- W2 day 0: edit the `subject` and `body` assignments in Build Welcome Email (Set node). They are expression strings, so `{{ $json.name }}` and friends stay evaluated.
- W2 day 1 / day 3, W3 day 0 / 3 / 7 / 14: edit `subject` and `text` in the email node itself.
- Any `{{ ... }}` you add is evaluated. Anything outside `{{ }}` is literal text.
- The email node's `text` field is plain text. To send HTML, open the node and use **Email Format -> HTML** with the HTML in the same field.
- Subject lines are prefixed with `=`, which marks the field as an expression. Keep the `=` when you rewrite a subject, or n8n stores your text literally and leaves `{{ }}` unresolved.

FAQ

1. Do I need n8n Cloud or can I self-host?

Either. Self-hosted Docker is the simplest for the Wait nodes, because a W3 run stays open for 14 days and you control execution retention. Cloud works too, with shorter retention limits and env var restrictions on lower plans. [verify] the Cloud limits for your plan.

2. Can I use Gmail, Outlook or a transactional provider?

Yes. The SMTP credential is provider-agnostic. Use an app password for Gmail and Outlook, or an SMTP

relay such as Postmark, Mailgun or SES. Send from a domain you control: a sequence sent from a free mailbox is likely to be filtered.

3. Why is the day 0 email in W2 a different node from the day 1 email?

The Set node builds the welcome email once, and the day 0 email forwards it. The later emails are written inline because they reference the Set node's values anyway. This keeps the welcome message - the one you rewrite most often - in a single location.

4. How do I get a score and summary instead of one integer?

That is already what W1 asks for: `<score>|<summary>`. If you rewrite the prompt, keep the pipe-delimited contract. The IF node parses the part before the pipe with `parseInt` and the alert body takes everything after it.

5. Can I make the onboarding sequence start automatically when a deal closes?

Yes, but not with the Manual Trigger. Replace Manual Start with a Webhook, Form Trigger or your CRM's node, and keep the field names `name`, `email`, `projectStart`, `deliverables`. Nothing else in W2 depends on the trigger type.

6. Does the pack store client data anywhere beyond n8n?

Email addresses and proposal text are stored in n8n's execution data, and the proposal text is also sent to your AI endpoint. Set `EXECUTIONS_DATA_PRUNE=true` with an `EXECUTIONS_DATA_MAX_AGE` that covers your longest sequence, or disable saving successful executions for W1 in the workflow settings. [verify]

Known limitations

- W2 is manual. A manual trigger has no production event, so onboarding does not fire by itself. See FAQ 5 for the three-node change.
- Webhooks are unauthenticated. Anyone with the URL can post a proposal or start a follow-up. Add webhook authentication (Basic, Header or JWT) before exposing the URLs publicly.
- No retry and no error branch. A failed HTTP call or SMTP error fails the run, visible in Executions. Enable node-level Retry On Fail or an Error Trigger workflow if you need it.
- The case study is a template. W3 day 7 contains no client results. Replace the paragraph with your own case study.
- W1 scores on a single integer. There is no structured breakdown in the alert; the raw model reply is included so you can see what it actually said.
- The pack is English only. Email copy, subjects and prompt are English (US).

Troubleshooting

A Wait node keeps the execution open for days - is that safe?

Yes, that is how the Wait node works. The execution is stored as waiting and the emails send when each interval elapses. For W3 that means a run stays open for about 14 days. Do not delete a waiting execution: deleting it cancels the remaining emails in that sequence.

I restarted n8n while executions were waiting. Did I lose them?

Waiting executions are persisted in the database and are picked up again by the instance after startup. [verify] on your version, and especially in queue mode with several workers: a worker that does not see the waiting execution will not resume it. The 60-second check is a wait node with `60` seconds in a scratch workflow, a restart before it fires, and a look at the execution status after startup.

Execution pruning killed my sequence

`EXECUTIONS_DATA_PRUNE=true` with a short `EXECUTIONS_DATA_MAX_AGE` (hours, default `336` = 14 days) deletes waiting executions before the last email sends. For W3 set it to at least `360`, or leave pruning off and prune manually. [verify] the current default on your build.

"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, and fill `From Email` (plus `To Email` on W1). W1 ships with an empty `To Email` on purpose, so a production run of W1 fails loudly until you fill it.

Import shows a node version warning

The file was saved with a newer node version than your n8n build supports. Upgrade n8n to 1.60 or newer and re-import. A warning is not fatal if the node still imports, but test the workflow before relying on it. [verify] the minimum version for your build.

"The file is not valid JSON" or the import silently does nothing

The file lost valid JSON, usually through a trailing comma or a straight quote turned into a typographic quote by an editor. Re-import the original file. JSON strings use `"`; expressions keep their inner single quotes. There are no comments in these files.

Webhook returns 404

Three usual causes: the workflow is not active (production URL), you call the production URL while the editor is only listening for a test event (use `/webhook-test/`), 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 `N8N_BLOCK_ENV_ACCESS_IN_NODE` is not `true`. On Cloud without environment variables, use the Header Auth credential described above.

OpenAI returns 404 or 400

The `url` must be the full chat completions path, `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 alert

Open the execution and look at the IF node output. A score below 7 takes the archive branch by design. A score of `0` means the model reply did not start with a digit - check the raw reply included in the archive branch, or in the HTTP node's output.

W1 alert arrives with empty fields

The webhook body did not contain those keys, or the sender omitted `Content-Type: application/json` and n8n could not parse the body. Missing fields render as empty strings by design.

Emails go out with the wrong greeting or a blank name

W3 reads `name` from the webhook body (`json.body.name`); W2 reads it from the Set node. If you

changed the webhook path, the trigger node name, or the Set node name, restore the node names the expressions reference - `$('Build Welcome Email')` and `$('Start Follow-Up (Webhook)')`.

"Expression error: cannot read properties of undefined"

A referenced node did not run, or a node name was changed. The expressions guard missing fields but do assume the trigger node and the Set node exist under their imported names.

The 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.