AGENTIC AI IN BUSINESS · WORKSHOP II



From request to deliverable

Morning 1 of 3 · today your mail AI assistant answers a mail you send · the Enron mail case

Frank Huettner · 9 November 2026

Today: six blocks and their outcomes

BEFORE THE BREAK: N8N, AND YOUR FIRST WORKFLOW

1

Why n8n, and the mail case

Outcome: What you will build today

2

WS2.1: your first n8n workflow

Outcome: Your Triage flow, tested on five mails

Break · 20 minutes

AFTER THE BREAK: YOUR MAIL AI ASSISTANT RUNS, AND YOU MAKE IT BETTER

3

Your mail AI assistant answers your mail

n8n workflows WS2.3 and WS2.4

Outcome: Kaminski's answer in your SKKU inbox

4

Challenge: all five test mails right

Outcome: Your score, and the best prompt in the room

5

Let's look inside WS2.3, and make it ours

Outcome: Your rule stars an urgent mail in Gmail

6

Why we put the human in the loop in WS2.4

Outcome: Why a person approves every draft

1

Why n8n, and the mail case

YOUR WINS

- [x] Your Triage flow in Langflow (homework)
- [x] Your test mailbox, loaded (homework)
- [ ] Your Triage flow, tested in n8n
- [ ] Your mail gets an answer
- [ ] Your best prompt
- [ ] Your own rule in Gmail
- [ ] You know why a person approves every draft

Why n8n, when you already have Langflow?

A Langflow agent answers when it is asked. A company needs more. It needs someone in the driver's seat.

A trigger starts it

- a new mail
- a form
- a chat
- a timer

Connects

- hundreds of apps, ready to use
- anything else by HTTP, like our Langflow

Checks, retries, human in the loop

- checks before it acts
- tries again when something fails
- human in the loop: waits for a person's OK

Runs and data, built in

- every run saved, each step visible
- a database for small to medium data sets

Langflow is the brain, n8n is the muscle

Each one does what it was built for.

Langflow: the brain

- came with the LLMs
- thinks in Python
- good for AI and data science: models, prompts, regression

n8n: the muscle

- more mature, built for business automation
- thinks in JavaScript
- good for the web and communication: mail, forms, other apps

Today n8n calls your Langflow flows: Langflow proposes, n8n checks and acts.

Who is behind n8n: the best of both worlds

1. **The company n8n GmbH, in Berlin, is behind n8n.**
2. **Its source code is public.**

You may run it yourself, for free, also in a company.

Future-proof

- a company maintains it
- no lock-in: your workflows are files, and you can run n8n yourself

Support

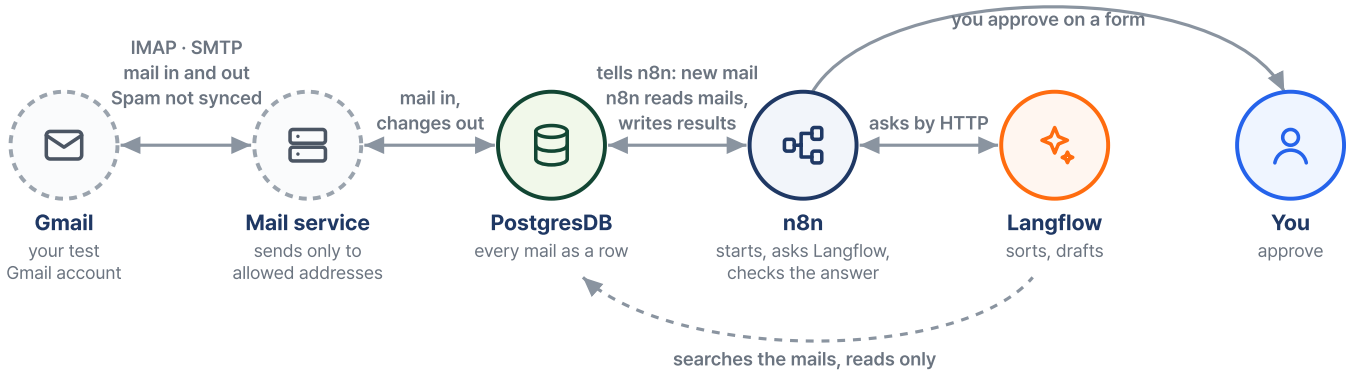
- professional support and hosting, for a fee
- or the community, for free

Self-hostable

- on your own computer, as today in kingo-pod
- then your data stays with you

The licence is "fair-code": free to use, also inside a company, but not to sell n8n itself as a service.

Why n8n talks to a database in our setup (not to Gmail as usual)



Any mail provider

- n8n has ready nodes for Gmail and Outlook
- we use plain mail (IMAP): it works with Naver Works too

PostgresDB in the middle: one place for all mail

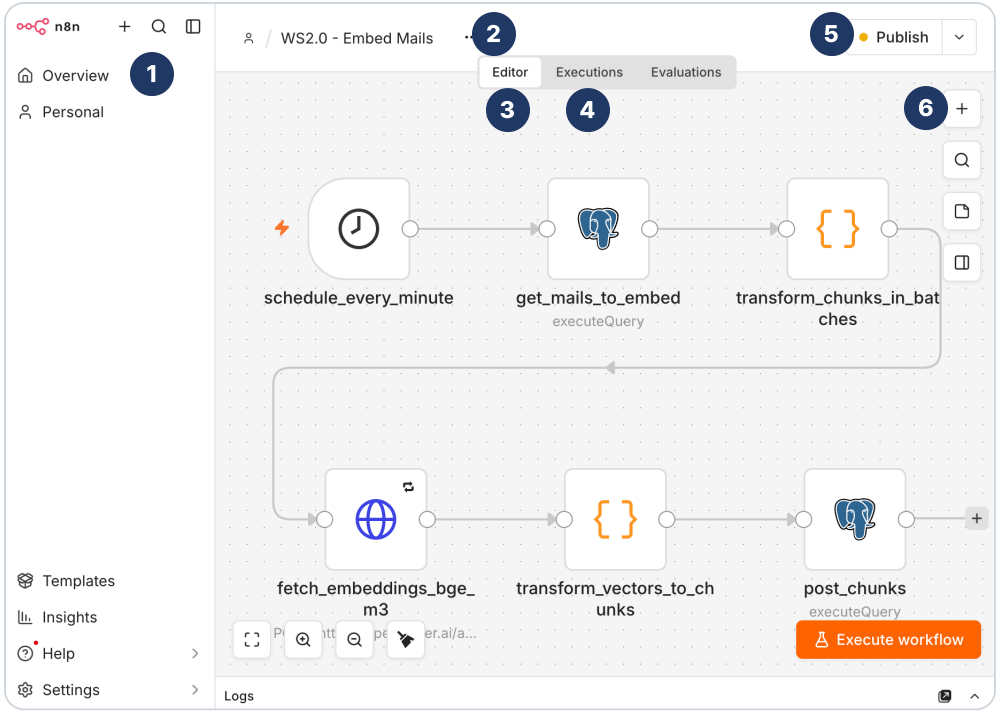
- every mail is a row: n8n reads and writes rows, not Gmail
- the search uses the same rows: no copy in ChromaDB

The mail service guards against accidents

- In n8n, Kaminski's contacts have real addresses
- any mail to them lands in your own test Gmail account

ALIGN YOUR TEAM AROUND PRINCIPLES FOR SUCCESS **One place for the data.** Ideally, you decide first where the data lives, and who may read or write it. Here: everything in PostgresDB; n8n writes, the Langflow flows only read.

The n8n editor: where things are



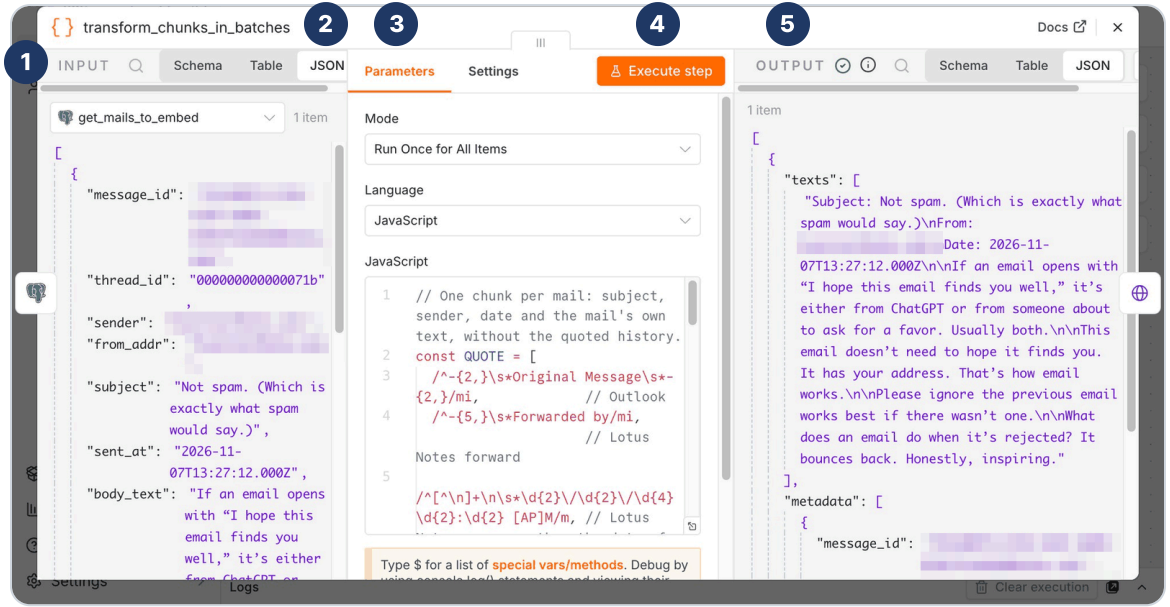
1. **Overview:** your workflows, credentials, executions and data tables, one tab each.
2. **··· next to the name:** Import, Download, Settings.
3. **Editor:** where you build the workflow.
4. **Executions:** every saved run of this workflow. Open one: each node with what came in and what went out.
5. **Publish:** from then on, the trigger works by itself. Green dot: live. Yellow dot: your changes are not live yet.
6. **+ on the right:** a new node (or the key `N`). `Shift` + `S`: a sticky note.

WS2.0, node by node: in the appendix.

Open n8n now: go to `localhost:5678`, click *Overview*, open `WS2.0 – Embed Mails`.

Double-click a node: in on the left, out on the right

n8n workflow `WS2.0 – Embed Mails`, node 3 `transform_chunks_in_batches`, in a run by hand*



1. **Input:** what came in from the node before, here `get_mails_to_embed`: 1 item, one mail.
2. **Schema, Table, JSON:** the same items, three ways to look at them.
3. **Parameters:** what the node does. Here JavaScript: one text per mail, in groups of 32.
4. **Execute step:** runs only this node, on the input on the left.
5. **Output:** what goes out to the next node: 1 item with `texts` and `metadata`.

* Your list of WS2.0's runs is nearly empty: in homework step 10.4 you set it to keep only failed runs. To see a run like this one: unpublish WS2.0, send a mail to your test Gmail account, click *Execute workflow*, then publish again.

Every n8n workflow: a trigger, then nodes

A trigger starts it

- A timer (e.g. WS2.0 - Embed Mails)
- An external event (e.g. WS2.3 - Triage New Mail)
- A page is opened (e.g. WS2.4 - Approvals Page)
- You click *Execute workflow* in the editor (e.g. WS2.1 - TEST - Triage)

Langflow runs when you call it.

Many items

- a list of items (an "array") goes from node to node
- WS2.0: one item per mail, up to 320 in a run
- each node works on every item of the list

Langflow sends one message at a time through the whole flow.

Left to right, top before bottom

- from the trigger to the last node
- when it splits: the upper branch runs first
- never back, unless you loop on purpose

Langflow does not care where a component sits: only the connections count.

Runs you can open

- double-click a node in a run
- left: what came in
- middle: what happens in the node
- right: what went out

Langflow shows the last run.

The case: Vince Kaminski's mailbox

- Kaminski ran the research group at Enron.
- His mailbox from spring 2001 is public (released after Enron's collapse at the end of 2001 by the US energy regulator (FERC) during its investigation).
- The emails and the names are real.
- In your test Gmail account every address ends in `.example`; n8n and Langflow see the real ones. No mail reaches these people.
- What comes in: questions, meetings, papers, students, hiring, and a lot of FYI.

Some mails are confidential, like resumes. The mail AI assistant may sort them, but never quote them.

3,244
real mails in your test Gmail account, January to April 2001

451
of them confidential: Draft Reply, the Langflow flow that writes drafts, cannot read them

140
test mails of May 2001, newer than your mailbox: five of them test your Triage flow today

Nobody appreciates cleanly sorted files, until they find themselves in chaos

n8n · your workflows at the end of today	
WS2.0 - Embed Mails	homework
WS2.1 - TEST - Triage	today
WS2.2 - Ask Mailbox	extra
WS2.3 - Triage New Mail	today
WS2.4 - Reply with Approval	today
WS2.4 - Approvals Page	today
WS2.5 to WS2.9	later

Search for “WS2.1” in the tabs Workflows and Data tables: the test and its two tables come up.

Workflow	`WS2.<step> – <purpose>` `WS2.3 – Triage New Mail`
A test	`WS2.<step> – TEST – <what>` `WS2.1 – TEST – Triage`
The test's data tables	`WS2.1 – TEST – Triage – Cases` `WS2.1 – TEST – Triage – Runs`
Files	small letters and _ `ws2_1_test_triage_cases.csv`
Nodes	what the node does, first `fetch_triage` · `post_result` · `if_confidential`
Your team	your team's name in place of `WS2.<step>` `Blue – TEST – Intake`

`Shift` + `S` in n8n: a sticky note with what the workflow does, your name, the date.

2

WS2.1: your first workflow

YOUR WINS

- ✓ Your Triage flow in Langflow (homework)
- ✓ Your test mailbox, loaded (homework)
- ● **Your Triage flow, tested in n8n**
- ○ Your mail gets an answer
- ○ Your best prompt
- ○ Your own rule in Gmail
- ○ You know why a person approves every draft

Ready? Three checks

- ✔ kingo-pod runs: in the terminal, in your kingo-pod folder, `./kingo up`. n8n opens at `localhost:5678`.
- ✔ Your test Gmail account is open in another browser tab.
- ✔ You downloaded `ws2_s1.zip` and unzipped it: the folder `ws2_s1` holds the workbook, these slides, two CSV files and the folder `n8n`.

Everything else you set up in the homework.

Before you build: two data tables

What is a data table?
A table that lives in n8n, like a small spreadsheet.

Why two?

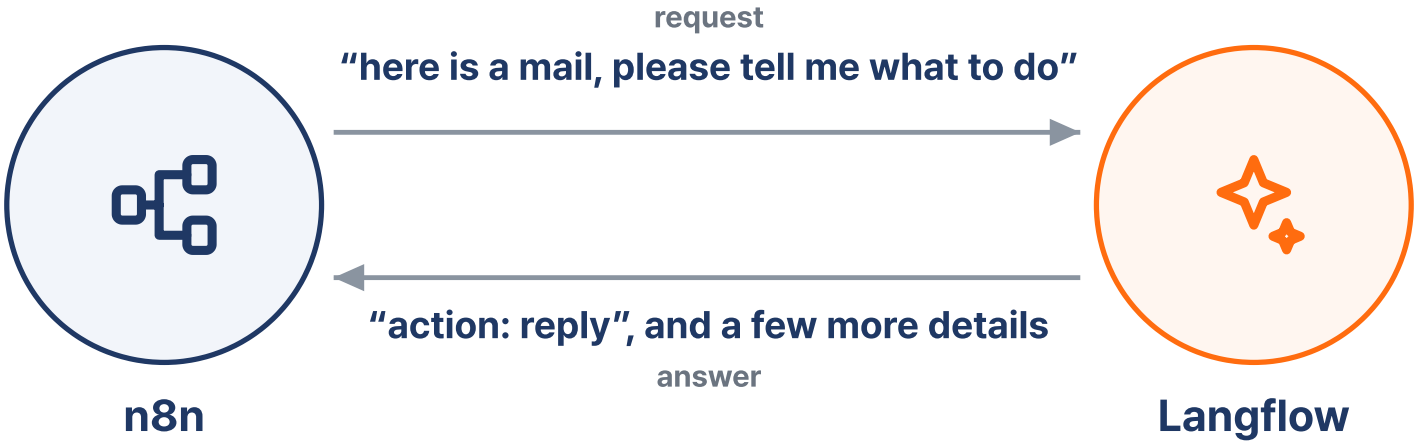
- *Cases*: the five test mails, each with the action expected
- *Runs*: empty for now; in block 4 it gets one line per test run

Important: everyone does this now, first. Your first workflow, WS2.1, reads its test mails from *Cases*. Without the two tables, it cannot run.

In the workbook: block 2, *Make the two data tables*, from the two CSV files in your folder `ws2_s1` .

n8n can reach out to Langflow with so-called HTTP requests

n8n sends the mail to your Langflow flow Triage and gets its answer back.



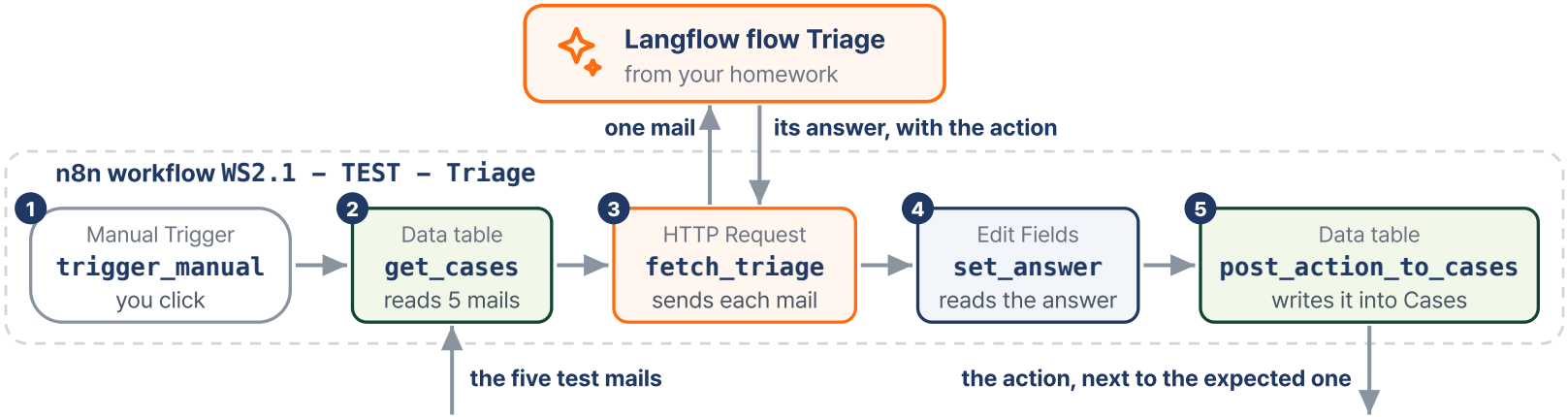
ALIGN YOUR TEAM AROUND PRINCIPLES FOR SUCCESS **Make a CONTRACT about what goes in, what comes out.** Ideally, you decide this before you build: what is the structure of the input to a receiving flow, and what is the structure of its answers. Here: the Langflow flow Triage gets one mail and answers with:

- *action*: calendar, reply, forward or archive
- *deadline*: as written in the mail
- *checks_ok*: did the answer pass the fixed checks?
- *events*: what the mail asks for or delivers, such as a meeting
- *why*: the reason for the action
- *problems*: what the checks found

n8n then reads exactly these.

Your 1st n8n workflow will test the Langflow flow Triage

How many of five test mails does your Langflow flow Triage sort right?



n8n data table
Cases

test_mail	subject	expected	action
8	Papers	reply	reply
30	(no subject)	forward	reply
46	Market Forecasting Conference	reply	reply
67	RE: Jump	archive	archive
126	Contact Info for Conference Call Tomorrow ...	calendar	calendar

ALIGN YOUR TEAM AROUND PRINCIPLES FOR SUCCESS **One task per piece.** Ideally, you give each piece one task. For example, the Langflow flow Triage only sorts. It doesn't reply or anything else. Breaking it down makes it easy to test and to reuse.

Build WS2.1: a trigger, then four nodes

1

Create

a new workflow, named `WS2.1 – TEST – Triage`

2

Add

a Manual Trigger, then the four nodes of WS2.1, one after the other

3

Run

Execute workflow. In *Cases*, Triage's answer stands next to the expected one.

In the workbook: block 2, *WS2.1*: every step, and every setting to copy.

✓ Done: your first n8n workflow calls your Triage flow and tests it.

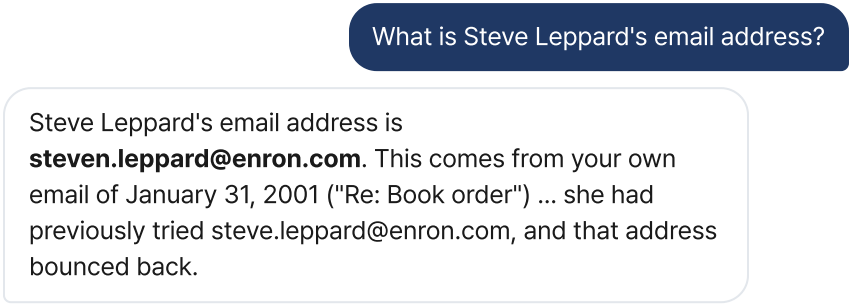
Extra, if you are fast: build WS2.2 - Ask Mailbox

A chat that searches your mail: the playground of homework step 13, now in n8n.

1 Chat Trigger `trigger_chat` → 2 HTTP Request → Langflow `fetch_ask_mailbox` → 3 Edit Fields `set_answer`

`trigger_chat`	your question, typed into the chat
`fetch_ask_mailbox`	asks the Langflow flow Ask Mailbox
`set_answer`	takes out the answer, for the chat

WHAT YOU SEE



In the workbook: block 2, *Extra: build WS2.2 - Ask Mailbox*: every step, and every setting to copy.

Extra, if you are fast: let an AI chat write the SQL

Who wrote to Kaminski most often? The answer is in PostgresDB. An AI chat writes the query in SQL, the language of databases: you do not need to know it.

1

Ask an AI chat

Copy the request from the workbook into ChatGPT, Claude or another AI chat.

2

Paste the query

into a Postgres node, in a new workflow `WS2.2 – Top Senders`.

3

Run and check

Execute workflow: ten senders. Does the list make sense? If not, tell the AI chat what is wrong.

The credential may only read and write what n8n needs, so a wrong query cannot break the mailbox.

In the workbook: block 2, *Extra: let an AI chat write the SQL*: the request to copy, and every step.

Break · 20 minutes

Your first workflow runs. After the break: your mail AI assistant answers your mail.

3

Your mail AI assistant answers your mail

YOUR WINS

- ✓ Your Triage flow in Langflow (homework)
- ✓ Your test mailbox, loaded (homework)
- ✓ Your Triage flow, tested in n8n
- ● **Your mail gets an answer**
- ○ Your best prompt
- ○ Your own rule in Gmail
- ○ You know why a person approves every draft

The path of your mail



Your SKKU mail
a question, to your test Gmail account

PostgresDB
the mail service copies it in, and n8n wakes up

WS2.3 - Triage New Mail
takes the mail, asks the Langflow flow Triage

WS2.4 - Reply with Approval
asks the Langflow flow Draft Reply for a draft, checks it

WS2.4 - Approvals Page
you approve

Your SKKU inbox
Kaminski's answer

1

Import

three ready workflows from the folder `n8n`

2

Connect

WS2.3 to WS2.4: WS2.3 hands every mail that needs a reply to WS2.4

3

Publish

both WS2.4 workflows first, then WS2.3: WS2.4 must already run when WS2.3 calls it

In the workbook: block 3: the three files, every click, and the address of the approvals page.

Try it: a mail from your SKKU account

your SKKU mailbox

Steve's address

You to your test address

Dear Vince, what is Steve Leppard's email address? I need to reach him.

later, after your approval

Vince J Kaminski to you

Kaminski's answer, drafted from his own mails, approved by you.

localhost:5678/webhook/approvals

Drafts waiting for you

To: your SKKU address
Subject: **Re: Steve's address**

The draft of Draft Reply (Enron), checked by WS2.4

Open the approval form

Send from the SKKU address you allowed in homework step 6, and ask a question. A mail that only informs is archived: no draft. The mail to copy: workbook, block 3.

✓ Done: your mail AI assistant answers mail, and you stay in control.

4

Challenge: all five test mails right

YOUR WINS

- ✓ Your Triage flow in Langflow (homework)
- ✓ Your test mailbox, loaded (homework)
- ✓ Your Triage flow, tested in n8n
- ✓ Your mail gets an answer
- ● **Your best prompt**
- ○ Your own rule in Gmail
- ○ You know why a person approves every draft

Five test mails: how many does yours get right?

✓ ×: our run of 2 Nov

8	30	46	67	126
Papers	(no subject)	Market Forecasting Conference	RE: Jump	Conference Call Tomorrow …
expected reply	expected forward	expected reply	expected archive	expected calendar
reply ✓	**reply ×**	**reply ✓**	**archive ✓**	**calendar ✓**

A ready workflow to import: `WS2.1 – TEST – Triage – Improved`. New, compared to your WS2.1:

1 · Settings

Which flow, how many rounds, which part of the answer to compare: in `set_test_settings`, or on its page.

2 · Rounds

Each test mail once per round, three rounds: one round can be lucky.

3 · One line per run

In *Runs*: what you changed, the score of each round, the seconds, every answer.

4 · A dashboard

Better than before? Which mail broke? A page where you set the test and run it.

3–4 / 5

Our rounds on 2 and 8 Nov, model GLM 5.3 Flash: 3 or 4 of 5, once 5.

Can you make it 5 in every round?

In the workbook: block 4: every step, and a question to check yourself.

Make it better: the prompt

In Langflow, open your flow **Triage (Enron)**, then its component **Prompt Template**. The actions are described there:

```
"calendar"  a meeting, call, visit or interview is
            asked for or set: to Shirley Crenshaw …
"reply"     someone wants information, an answer
            or a decision from Kaminski …
"forward"   someone wants something done …
```

In Langflow: change only the prompt, then save the flow. On the page of `WS2.1 – TEST – Triage – Improved`: write what you changed, then *Run the test*.

- Read test mail 30: its whole text is in the column `mail` of the data table *Cases*. Why does the model say `reply`, not `forward`?
- Who has 5 of 5 in all three rounds? Read the prompt aloud.
- Did your fix for one mail break another?
- Is five enough to tell a better system from a worse one?

Test your project's Langflow flows from n8n, too

- Copy `WS2.1 – TEST – Triage – Improved`, give it your test cases and your flow: it tests every Langflow flow of your team's project.
- Run it again after every change, and compare in its dashboard.
- It also avoids the Langflow 1.8 memory bug. In the Playground, chats with the same name share one memory, even across flows: one test can remember the one before. From n8n, every call of the test starts with an empty memory, a session of its own.

In the workbook: block 4, *Test your project's Langflow flows from n8n, too.*

✓ Done: you test your Triage flow the right way: three rounds, one line per run.

5

Let's look inside WS2.3, and make it ours

YOUR WINS

- ✓ Your Triage flow in Langflow (homework)
- ✓ Your test mailbox, loaded (homework)
- ✓ Your Triage flow, tested in n8n
- ✓ Your mail gets an answer
- ✓ Your best prompt
- ● **Your own rule in Gmail**
- ○ You know why a person approves every draft

What WS2.3 did with your mail

Your mail from block 3, “Steve's address”, went through five steps in WS2.3.

1

Wake up

Your mail lands in PostgresDB. PostgresDB tells n8n at once, and WS2.3 starts. Just in case, WS2.3 also looks every 15 minutes.

2

Take it once

WS2.3 notes the mail as taken. No other run handles it again, so it never gets two answers.

3

Protect

Card and bank numbers are masked before a model sees the mail. A mail that tries to give the assistant orders goes to a person.

4

Ask Triage

Your Langflow flow from the homework reads the mail and proposes what to do.

5

Act

A fixed rule first. Then the action in Gmail: a label, the archive, or a draft for your OK.

Open WS2.3, tab *Executions*, the run of your mail. Double-click each node: what came in, what went out.

In the workbook: block 5, *Look inside WS2.3*: each step and its nodes.

Triage proposes an action, n8n checks it, then acts

A model answers with text, and text can come out broken. So n8n checks before it acts.

The answer of your Langflow flow Triage to “Steve's address”, in Frank's run:

```
{
  "action": "reply",
  "why": "Frank Huettner asks Kaminski for Steve
Leppard's email address, so a reply with the information
is needed.",
  "events": ["Request_Data"],
  "deadline": "",
  "checks_ok": true,
  "problems": []
}
```

- `action`: what to do. One of four: `reply`, `calendar`, `forward`, `archive`.
- `checks_ok`: is the answer in order? The component Check Triage in your flow checks it: one complete answer, with one of the four actions.
- Whether the action is right, the check cannot tell. Your test in block 4 measures that.

In order

n8n does what `action` says.

Not in order

No action. The mail gets the label *Assistant/Check*, and a person looks at it.

What WS2.3 does in Gmail: a rule first, then Triage's action

Your test Gmail account, inbox, after WS2.3			Triage's action: what WS2.3 did
☆	**Dorothy Dalton**	Assistant/Confidential message from Ken Rice	marked confidential: the rule, whatever Triage said
☆	**Jason Sokolov**	Papers	reply: WS2.4 drafts an answer, you approve it
☆	**lopez**	Assistant/Calendar Risk Systems Enhancement Mtg - EB50M03	calendar: a label
★	**Lon Draper**	Assistant/Forward Strip of Spread Options	forward: a label and a star
☆	~~Stinson Gibner~~	~~Jury Duty~~	archive: out of the inbox, not deleted
☆	**any sender**	Assistant/Check *an answer not in order, or a mail that gives orders*	not in order: a person looks at it

ALIGN YOUR TEAM AROUND PRINCIPLES FOR SUCCESS **What a rule decides, not the model**

- **Ideally:** before you build, decide which decisions a fixed rule makes. A rule decides the same way every time, and you can check it.
- **Here:** the mailbox marks some mails as confidential, such as a resume or an evaluation. Such a mail only gets its label, never a draft, whatever Triage proposes.

n8n never touches Gmail itself: it writes one row into the table `changes`, such as “star this mail”. The mail service does it in Gmail. Your own rule, next, works the same way.

Your turn: a rule of your own

In `WS2.3 – Triage New Mail`, a second branch from `check_triage`



`if_urgent` asks: does the subject say "urgent"? If so, `post_star` gives the mail a star in Gmail.

In the workbook: block 5, *Your own rule*: how to add the branch, and the settings to copy.

Publish WS2.3 again, then send “Urgent: call me back” from your SKKU account:

★	**You**	Urgent: call me back	your rule

✓ Done: your own rule, running in a real system. A rule decides here, not the model: the same every time.

6

Why we put the human in the loop in WS2.4

YOUR WINS

- ✓ Your Triage flow in Langflow (homework)
- ✓ Your test mailbox, loaded (homework)
- ✓ Your Triage flow, tested in n8n
- ✓ Your mail gets an answer
- ✓ Your best prompt
- ✓ Your own rule in Gmail
- ● **You know why a person approves every draft**

Often, a little adjustment is all you need

The mail AI assistant's draft named the right person, as Kaminski did.

Incoming mail · **From:** John A. Cote · 11 May 2001 · **Subject:** UT Recruiting

Vince, Czesc,
Our group would like to recruit a new hire / intern from UT's energy studies program. Kristin thought you might have a contact in this program. (Any info would be helpful!) Dziekuje.

Kaminski's actual response · 15 May 2001

John,
You can contact Prof. Ehud Ronn.
Ehud I. Ronn, Professor of Finance, Director, Center for Energy Finance Education and Research, University of Texas at Austin · phone · mail
Vince

Draft of the mail AI assistant · the Langflow flow Draft Reply

John,
Czesc. Good to hear from you.
I do not have a contact in the energy studies program itself.
My contacts at UT Austin are Ehud Ronn and Sheridan Titman
... I will ask them about the program ...
Vince

This draft saved some time. Possible extension: another loop that incorporates your adjustments.

Yanbo's five trust components, in your mail AI assistant

Component	Principle behind it	In your mail AI assistant
Input validation	What a rule decides	Before a model sees a mail, WS2.3 masks card and bank numbers. A mail that tries to give the mail AI assistant orders goes to a person. WS2.4 masks again before Draft Reply.
Process guardrails	What each part may do	The Langflow flows may only read the mailbox. Draft Reply never sees a confidential mail: the database hides it. Only your OK lets a mail go out.
Output filter	Make a contract about what goes in, what comes out	Each Langflow flow checks its own answer by fixed rules. Check Triage: one of the four actions. Check Draft: a greeting, no gap like [Name], addresses and numbers only from the mails. n8n acts only on an answer in order. **You** see Check Draft's result on the approval form.
Error handling	Where a person decides	n8n calls Langflow up to twice. Triage's answer not in order: a person looks at the mail. No draft, or no one to send it to: nothing is proposed, and the problem is noted.
Audit trail	One place for the data	n8n's *Executions*: every run that decides, node by node. The table `outbox`: each draft with its status (proposed, approved, sent), who approved it, and why.

- The middle column: the principle behind each component. All six principles: in the appendix.
- As with Yanbo, each Langflow flow checks its own answer. n8n checks the input and decides what happens.

What you built today

- ✔ A mail AI assistant that answers your mail, after your OK
- ✔ A better prompt for your Triage flow, measured in three rounds
- ✔ Your own rule: an urgent mail gets a star in Gmail

> The big idea: the model proposes, a rule checks, a person approves.

Appendix: six principles to align your team around

You saw each one today. Write down your team's own, before anyone builds.

ABOUT THE PIECES

1 · One task per piece
The Langflow flow Triage only sorts: easy to test (WS2.1) and to reuse.

2 · Make a contract about what goes in, what comes out
The Langflow flow Triage: fixed fields. n8n reads exactly these.

3 · One place for the data
The mails and the search in PostgresDB. n8n writes, Langflow reads.

ABOUT CONTROL

4 · What a rule decides
Not the model: a confidential mail gets a label, never a draft.

5 · What each part may do
Draft Reply cannot send. Only your OK lets a mail go out.

6 · Where a person decides
Nothing leaves without your OK. WS2.4 waits on a form.

For your project: the same six in `ws2_rules.html` in your download, with your team's spec at its end. A team that agrees on them first spends its time on the project, not on making its pieces fit.

Appendix: one contract that both tools can read

n8n and Langflow do not tell each other what a flow needs and gives. The standard way: write it down once, in a file, and check both sides against it.

JSON Schema

Describes one JSON object: which fields, which type, which values. One file per flow, and Langflow, n8n and the tests all check against it. Yanbo's "structured output with a schema" is the same idea.

MCP (Model Context Protocol)

For AI agents that choose their tools live, while they work. Each tool comes with a name, what it does, and a JSON Schema for in and out. Natural for us: a Langflow agent calls n8n workflows as tools. A fixed step, like n8n calling Triage, needs no MCP.

All four flows: `contracts/flows.openapi.yaml` (OpenAPI with JSON Schemas). Not needed in the course. Using a coding agent anyway, such as Claude Code? Ask it to write yours: the prompt is in the workbook.

Appendix: WS2.0 from your homework, node by node

n8n workflow `WS2.0 – Embed Mails`: makes new mails searchable, every minute

1. Schedule Trigger
   `schedule_every_minute`
   starts it every minute
2. Postgres
   `get_mails_to_embed`
   the mails not yet searchable
3. Code
   `transform_chunks_in_batches`
   one text per mail, in groups of 32
4. HTTP Request → OpenRouter
   `fetch_embeddings_bge_m3`
   one request per group
5. Code
   `transform_vectors_to_chunks`
   each embedding back to its mail
6. Postgres
   `post_chunks`
   saves them in PostgresDB

Node 3: why one text per mail

- subject, sender, date and the mail's own words
- the quoted older mails cut off: they are stored as mails of their own

Node 3: why groups of 32

- node 4 sends one request to the model for 32 mails, not 32 requests
- fewer and faster calls; a failed call repeats only its group