BUILD WITH IT

Agents and Automation

What an agent actually is, and when you should build a script instead.

First edition, September 2026

Addaly

Series editor: Dr Mustafa Mun

Draft, open for academic review

BUILD WITH IT

Agents and Automation

What an agent actually is, and when you should build a script instead.

Series editor: Dr Mustafa Mun

Addaly

First edition, September 2026 · Draft, open for academic review

Agents and Automation

First edition, September 2026, build 62a26075.

Written by AI models to a written standard, with Dr Mustafa Mun as series editor. Exam keys checked blind by a second model: 3,665 of 3,666 agreed. Academic review invited.

The manuscript was drafted with the assistance of a large language model and was edited and published under his name. That is stated plainly here rather than left to be discovered, because a reader is entitled to know how a book they are trusting was made.

How to cite: *Mun, M. (2026). Agents and Automation. Addaly. addaly.com/learn/agents-and-automation.*

This book is the printed form of the course of the same name at addaly.com/learn/agents-and-automation. The website is the living version and is corrected first. Where the two differ, the website is right.

Free to read, free to download, free to print, and free to teach from. There is no paid edition and there never will be.

Every diagram in it is drawn from data by code, not generated as a picture, so the numbers on the page are the numbers in the argument. The answers to every check, test and examination question are at the back.

8 modules · 72 lessons · 27 figures · 8 case studies · 50,354 words · about 11 hours

Brief contents

1

MODULE 1

Before you build: script, workflow, or agent

Given any automation request, choose between a script, a fixed workflow and an agent, and justify the choice with that shape's per-run cost, failure rate and recovery story.

Lesson 1 · 6 min

Three machines, one word

THE ONE THING TO KEEP

An agent is a program where the model, not the author, decides what happens next.

The question that separates them

Three different machines get called "an agent." One question tells them apart: **who decides what happens next?**

A chatbot: you decide. You type, it answers, you type again. Every step forward comes from a human, and it never touches anything outside the conversation. A model with no tools connected is a chatbot. That is genuinely useful, and it is not an agent.

A workflow: the author decided, before it ever ran. Somebody drew a graph — trigger, step, branch, step — and the graph is the same every time. A Zapier Zap, an n8n flow, a Make scenario, a cron job running a Python script. The path is fixed at build time.

Here is the part that trips people up: **a workflow with a model in it is still a workflow.** Take this n8n flow, running at a coaching centre in Bengaluru:

```
new WhatsApp message
  → model step: enquiry, fee question, or other?
  → if enquiry:  send brochure, add row to sheet
  → if fee:      send fee schedule, notify office
  → else:        forward to a human
```

There is a language model in the middle making a judgement call. It is still not an agent. Every path that flow can ever take was drawn by a person in advance. The model is filling in one blank on a form somebody else designed.

An agent: the model decides, at run time, in a loop. You give it a goal, some tools, and no fixed path. It picks a tool, sees the result, picks again. The tell is this: **you do not know how many steps it will take before it runs.** Same goal, different input, different number of steps.

The agent version of the same coaching centre: "Here is the WhatsApp inbox, the fee structure, the batch timetable, and the ability to book seats. Handle enquiries." Now the work depends on what the parent asks. A simple question might take one step. A parent negotiating a transfer between batches, checking two timetables and a refund rule, might take nine.

It is a dial, not a switch

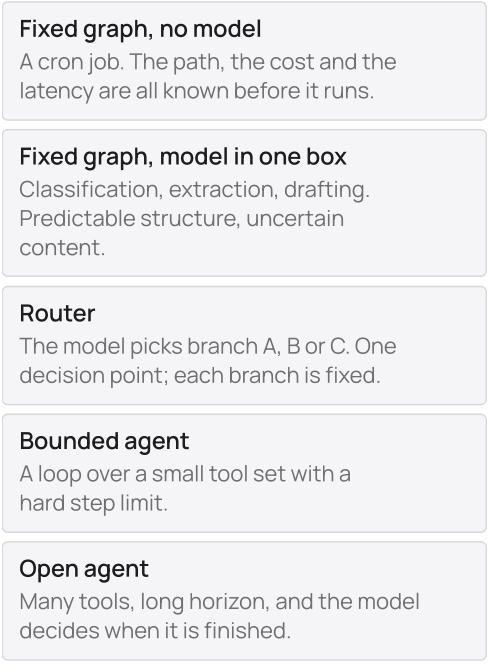
Most real systems sit somewhere in between, and it helps to say where.

- **Fixed graph, no model.** A cron job. Fully predictable.
- **Fixed graph, model in one box.** Classification, extraction, drafting. Predictable structure, uncertain content.
- **Router.** The model picks branch A, B or C; each branch is fixed. One decision point.
- **Bounded agent.** A loop over a small tool set with a hard step limit.
- **Open agent.** Many tools, long horizon, the model decides when it is finished.

Move down that list and you gain flexibility. You also lose, in this order: predictable cost, predictable latency, testability, and your ability to explain to a customer why it did what it did.

Figure 1.1

The dial, from a cron job to an open agent



Going down the list you gain flexibility and lose, in this order: predictable cost, predictable latency, testability, and your ability to explain to a customer why it did what it did.

Why the vocabulary is worth getting right

Almost everything currently sold as "an AI agent" sits in the middle of that dial. That is usually good news rather than a criticism. Workflows cost roughly the same every run, you can test them, and when one breaks you can point at the box that broke.

The mistake that costs people months is reaching for the bottom of the dial first. Someone builds an open agent for a task that was always going to take the same seven steps. It works in the demo, costs forty times what the workflow would have cost, fails in a new way each week, and nobody can debug it because "it decided to."

So when a task lands on your desk, ask the question first: **does the sequence of steps actually change from run to run?** If it does not, you want a workflow, possibly with a model inside one of its boxes. If it genuinely does — if the shape of the work depends on what you find partway through — then you want an agent, and the rest of this course is about what that costs you.

BEFORE YOU MOVE ON

An n8n flow receives a support email, uses a model to sort it into billing, technical or other, routes it into one of three fixed sub-flows, has a model draft a reply, and sends it. No human touches it. Is this an agent?

A. No — every path it can take was drawn before it ran; the model only picks between branches somebody else authored.

B. Yes — a language model is making the decisions, and decision-making is what makes something an agent.

C. Yes — it takes real actions in the world and sends email without a human approving anything.

D. No — it uses the model for classification and drafting rather than through a tool-calling API.

The answer and why: p. 383

Lesson 2 · 8 min

The boring stack that already does this

THE ONE THING TO KEEP

Most automation requests are a scheduler and thirty lines of code; put a model only in the one step where the input is genuinely unstructured.

Walk the request backwards

Someone asks whether AI can automate the Monday report. Before answering, find out what the report is. Usually: three numbers pulled from a database, arranged in an email, sent at six on Monday morning.

There is nothing in that description a model does better than a `SELECT` and a scheduler. It will do it more expensively, more slowly, and with a small chance of inventing a number.

This is not a reason to avoid models. It is a reason to find the step that actually needs one.

The five fields, and the trap in them

```
0 6 * * 1   /usr/bin/python3 /opt/reports/weekly.py
```

Minute, hour, day-of-month, month, day-of-week. That line means 06:00 every Monday.

Here is the part that catches people: when you set **both** day-of-month and day-of-week, cron treats them as OR, not AND. `0 6 13 * 5` runs on the 13th of every month *and* on every Friday. Most people read it as "Friday the 13th" and write a job that fires roughly nine times more often than they intended.

Four parts, and only one of them is clever

Nearly every automation is:

1. **A trigger** — a schedule, a webhook, a new row, a person pressing a button.
2. **A fetch** — an API call, a query, a file read.
3. **A transform** — the actual work.
4. **A delivery** — an email, a message, a row written back, a file saved.

Steps 1, 2 and 4 have had good deterministic answers for thirty years. Step 3 is where you decide.

Where a model earns its place

Put a model in the transform when the input is genuinely unstructured and varied:

- free text a human wrote, which needs classifying, summarising or extracting
- a document whose layout changes between senders
- a decision that depends on meaning rather than on a field

Everything else — arithmetic, formatting, routing on a known value, sending — stays as code. A pipeline with one model call in the middle is cheaper to run, easier to test, and fails in ways you can read.

The failure the boring stack has, and its mechanism

Deterministic pipelines break silently when the shape of the data changes, and the mechanism is almost always a default value.

```
total = row.get("total", 0)      # silent: the report says
revenue is zero
total = row["total"]             # loud: KeyError, and you find
out tonight
```

Upstream renames `total` to `total_amount`. The first line keeps running and reports zero revenue every night. The second stops and pages you. **Never default a missing field to a plausible value.** A crash is a message; a zero is a lie.

Add one sanity assertion at the end of the transform — this week's total is within some sane multiple of last week's — and you catch the entire class of failure that no exception will ever throw.

The free path

You do not need a platform to start.

- **cron**, on any Linux box you already pay for.
- **systemd timers**, if you want a missed run to catch up after a reboot: `OnCalendar=Mon --* 06:00:00` plus `Persistent=true`. Cron simply skips a run it slept through.
- **GitHub Actions** `schedule:` — free minutes on public repositories, five-minute minimum granularity. Be aware that scheduled workflows get disabled automatically after around 60 days without repository activity, and that runs at popular times drift by minutes.
- **Cloudflare Workers cron triggers**, if the work is short and lives at the edge.

Job adverts name **Zapier** and **Make**, and you should know them: both are excellent at connectors and both meter you per task or per operation. **n8n** self-hosted, **Node-RED** and **Windmill** do the same job for the price of a small server, and n8n has its own cloud if you would rather not run it.

Start with the boring one. You can always add a model to step 3 later; it is much harder to take an agent apart once a business depends on it.

BEFORE YOU MOVE ON

A nightly job pulls yesterday's orders and emails a revenue total. The upstream team renames the `total` column to `total_amount`. The job keeps running, and the email reports revenue of zero every night for nine days before anyone notices. Which change would most directly have surfaced the break on night one?

A. Read the field with `row['total']` instead of `row.get('total', 0)`, so a missing column raises instead of quietly producing a plausible number.

B. Add a model to the pipeline to work out which column now holds the total.

C. Move the job from cron to a workflow tool with a retry policy.

D. Run the job hourly instead of nightly so problems surface sooner.

The answer and why: p. 384

Lesson 3 · 9 min

Watch the process before you automate it

THE ONE THING TO KEEP

The process on paper is not the one that runs, and the ceiling on any automation is the share of elapsed time held by the steps it actually replaces.

Two versions of every process

There is the process as written down and the process as performed, and they are never the same document. A thirty-person firm in Coimbatore had a written invoice procedure with six steps. Sitting beside the person who ran it for one morning produced nineteen distinct actions, including a WhatsApp thread to ask which cost centre a delivery belonged to, and a spreadsheet that appeared in no procedure and no system diagram but decided which invoices were paid that week.

CASE STUDY · MODULE 1

Nineteen steps, and the spreadsheet nobody mentioned

An illustrative case, built from documented patterns.

A thirty-one person precision engineering firm in Coimbatore, deciding whether to build an invoice agent six weeks before the annual audit

Sundar Precision machines components for two automotive suppliers. Invoices from its own suppliers arrive as email attachments, about two hundred a week, and Meena — who has run accounts payable for nine years — types them into the accounting package by hand.

The finance head, Ravi, spent an evening with a hosted model, pasted in four scanned invoices, and got four clean JSON objects back. He came in on Monday wanting an agent: access to the mailbox and to the accounting system, told to keep the ledger up to date. A contractor quoted six weeks. The firm could afford it once.

Meena's objection was not about the technology. "Sit with me on Thursday," she said.

He did, and wrote timestamps rather than asking questions. The written procedure had six steps. Thursday produced nineteen distinct actions, and the two that consumed the most time were in no document anywhere. One was a WhatsApp thread with the stores supervisor asking which cost centre a delivery belonged to, because about a third of invoices arrive with no purchase order number. The other was a spreadsheet on Meena's desktop, absent from every system diagram, which decided which invoices got paid that week.

The arithmetic changed as he watched. Two hundred invoices at four minutes each is eight hundred minutes a week, which was the number in his proposal. But of the four minutes, the typing was about two and a half. The rest was the phone call, the message and the wait. On roughly thirty per cent of invoices, at least one of those was needed.

So the clean path — a hundred and forty invoices at two and a half minutes — is three hundred and fifty minutes. Reviewing what a machine did would cost perhaps fifteen minutes a day, which is another seventy-five. The honest saving was around two hundred and seventy-five minutes a week, not eight hundred. Still worth having. A third of what had been promised on Monday.

That left two proposals on the table, and they were not the same shape.

The contractor's version was an agent: an inbox tool, a PDF tool, a ledger tool, and an instruction to keep things up to date. It would handle the exceptions, or attempt to. It could chase a missing purchase order number, look up the supplier, guess a cost centre from last month's pattern. Six weeks, and a per-run cost nobody had estimated because nobody had counted the steps.

Meena's version, which she did not know was a proposal, was smaller. Change the supplier onboarding form so the purchase order number is a required field with a format check. Then a fixed pipeline: filter attachments from known supplier domains, one model call to extract eight fields, five lines of arithmetic to check that the line items sum to the subtotal and the subtotal plus tax equals the total, then either a row in the ledger or a review queue showing the PDF beside the extracted values. Three days of contractor time, and the form change was an afternoon.

The cost on Ravi's side of the argument was real. The small version does nothing at all about the thirty per cent, which is where Meena's week actually goes. It saves the typing and leaves the phone calls, and phone calls are what she would like to stop making. Choosing it means telling her, honestly, that the part she hates is the part staying.

The cost on Meena's side was also real, and it was the auditor. In six weeks somebody would ask how a particular invoice reached the ledger with that cost centre against it. "The system decided" is not an answer, and neither is a twenty-step transcript. She had watched a previous manager lose two days to a reconciliation query, and she was not prepared to own something she could not explain in a sentence.

Ravi asked the contractor one question that settled it: what does one run cost, and how many steps does it take. The contractor did not know, because the number of steps depends on the invoice. That was the definition from the first lesson of the course Ravi had half-finished, and it arrived at exactly the wrong moment for his proposal.

YOUR ANSWERS

1. Ravi's proposal was based on eight hundred minutes a week. Where did that number go wrong, and what should he have measured instead?

2. The form change removed half the missing purchase order numbers and involved no model at all. Why is that easy to miss when a project is framed as an automation project?

3. The contractor could not say how many steps one run would take. Why was that a decisive answer rather than a minor gap in the quote?

STOP HERE

Write your answers before you turn the page. How it played out starts on p. 48.

CASE STUDY · MODULE 1

How it played out

They built the small version, and changed the form first. The purchase order field became mandatory with a format check, which removed about half the missing-PO exceptions within a month — no model involved, and the cheapest improvement in the project.

The pipeline took four days rather than three. Extraction ran at ninety-six per cent on the total and seventy-one per cent on the due date, measured on a hundred hand-checked invoices, so due dates below a confidence threshold went to review with the page image and the quoted line highlighted. Meena's review took about twelve seconds each rather than four minutes.

Measured over eight weeks, the saving was two hundred and forty minutes a week, close to Ravi's honest estimate and nothing like his first one. The phone calls remained. Meena said the difference was that she now made them in one batch at eleven rather than all day.

The auditor asked about three invoices. Each answer took under a minute, because the pipeline stored the original PDF, the page number and the quoted line for every extracted field.

Eighteen months later they added an agent, for one thing only: chasing missing purchase order numbers across the stores WhatsApp export and the delivery notes. It genuinely takes a different number of steps each time. It drafts a message; a person sends it.

The questions, discussed

1. Ravi's proposal was based on eight hundred minutes a week. Where did that number go wrong, and what should he have measured instead?

Eight hundred minutes was the total elapsed time of the process, not the share of it the automation could replace. Only the typing — about two and a half of the four minutes — was automatable, and that only on the seventy per cent of invoices with no exception. The ceiling on any automation is the share of elapsed time held by the steps it actually removes. He should have measured,

per step: what arrives, what decision is made, how long it takes, how often it happens, and how often it goes wrong. He should also have subtracted the review time the automation creates, which was seventy-five minutes a week here.

2. The form change removed half the missing purchase order numbers and involved no model at all. Why is that easy to miss when a project is framed as an automation project?

Because the framing puts the work at the step where the pain is felt rather than at the step where it is caused. Meena spent her time chasing missing numbers, so the obvious target is the chasing. The number was missing because the form let people leave it blank. Fixing the form removes the work rather than automating it, which is cheaper, permanent, and does not need a maintainer. The general habit is to look one step upstream of the exception before designing anything: a share of exceptions is usually manufactured by an input that permits them.

3. The contractor could not say how many steps one run would take. Why was that a decisive answer rather than a minor gap in the quote?

Because it is the definition of an agent rather than a workflow: you do not know how many steps it will take before it runs. That single fact carries the consequences the firm cared about. Cost per run is unpredictable, because the transcript is re-sent every step and total spend grows roughly with the square of the step count. Latency is unpredictable for the same reason. And explainability is unpredictable, which mattered because an auditor was arriving in six weeks and "it decided to" is not an answer. The unknown step count was not a missing line in a spreadsheet; it was the whole risk profile of the proposal.

MODULE 1 TEST

Check what stuck

6 questions, one right answer each. This one is for you, not for a record. The answers, and the reason behind each, are in the key on p. 382. If two go wrong, re-read the module before the exam.

1 An n8n flow classifies each incoming WhatsApp message with a model call, then branches to one of three fixed paths. Why is this still a workflow rather than an agent?

A. Because n8n is a visual tool, and an agent can only be built in code with a framework behind it

B. Because every path it can take was drawn by a person before it ran; the model fills one blank on a form

C. Because a single model call cannot be an agent: an agent needs at least two different models cooperating with one another on the same task

D. Because it keeps no memory between messages, and memory is what turns a workflow into an agent

2 A nightly job reads three input files. A supplier renames one, so that file stops arriving. Which line turns this into six weeks of a plausible, wrong report?

A. total = row.get('total', 0), because a missing field becomes a plausible number and nothing raises

B. total = row['total'], because the unhandled KeyError leaves the report half written and half sent

C. A retry with exponential backoff, because it keeps trying the old filename until the job times out and skips the file entirely

D. An assertion comparing this week's total against last week's, because an assertion that fires stops the run

3 A process takes 800 minutes a week. Automation removes 2.5 of the 4 minutes per case, 30 per cent of cases still need a phone call, and reviewing the output costs 75 minutes a week. What is the honest saving?

A. 800 minutes, because the exceptions were going to happen anyway and are not the automation's fault

B. About 350 minutes, because the clean path is what is being automated and review is somebody else's budget line

C. About 275 minutes, because only the automatable share of the clean cases counts and review time is subtracted

D. Roughly 560 minutes, since seventy per cent of the total elapsed time sits on the clean path and that is the part being replaced

4 Sorting work by volume and by how much the inputs vary, which quadrant is the classic failed automation project?

A. High volume, low variance — invoices from the same twelve suppliers every month

B. High volume, high variance — support emails written by strangers

C. Low volume, low variance — a quarterly reconciliation whose shape is identical every single time it is run

D. Low volume, high variance — runs four times a year and looks different each time

5 A ten-step agent sends about 60,500 input tokens per run. The same agent doing the same job in five steps sends about 19,000. Why is the drop steeper than halving?

A. Shorter runs use fewer tools, and tool definitions are the largest part of every request

B. Providers charge a higher rate per token once a request passes a size threshold, so long runs pay a penalty rate

C. The transcript is re-sent every step, so the growing part of the cost scales with the square of the step count

D. The model produces more output tokens on longer runs, and output tokens are priced several times higher than input

6 A team is choosing between a hosted visual tool and a Python file. Which fact about the pricing should shape the design before anything is drawn?

A. Both bill per run of the whole flow, so the number of steps inside it does not affect the bill

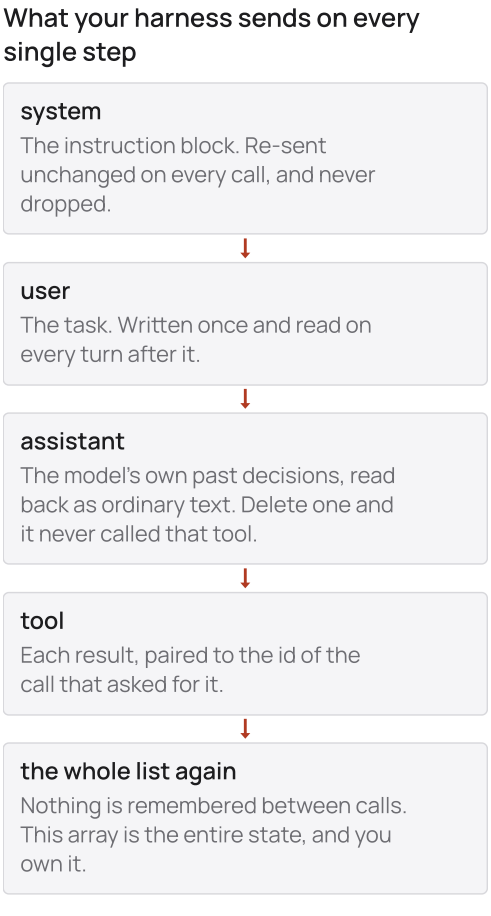
B. Both bill per task or operation, so a twelve-step flow run a thousand times bills twelve thousand units

C. Both bill by the volume of data moved between connectors, so a flow passing large payloads costs far more than one passing small ones

D. Both charge a flat monthly fee regardless of volume, so the design should optimise for developer time rather than run count

Four messages, and the model reads all four before deciding anything. Three things in there are worth noticing.

Figure 2.1



There is no session. Drop a tool result while trimming and leave its call behind, and the provider returns a 400 rather than degrading gracefully, so trim in whole pairs.

The pairing is load-bearing. Every `tool_call_id` must match a preceding tool call. Drop one while trimming history and the provider returns a 400, not a graceful degradation. Trim in whole pairs.

The model reads its own past decisions as text. That assistant message is not a memory; it is a transcript entry. If you delete it, the model has no idea it ever called `get_order`.

Some providers hand back reasoning blocks and expect them returned in the next request. If your harness quietly strips them, the model loses its own train of thought between steps, and the symptom is an agent that seems to restart its thinking every turn.

Consequence one: history is editable, and that is your best tool

Since you own the list, you can rewrite it. Legitimately and routinely:

- **Drop a failed detour.** Three steps of a dead end, replaced by one line: *tried the invoices API, it has no refund endpoint.*
- **Replace a huge result with a summary.** The 30,000-token document becomes a 200-token abstract plus a file path.
- **Inject a note as a tool result.** *The user has been waiting 40 seconds; prefer finishing over further searching.*
- **Re-state the goal near the end**, where it will actually be read.

The model cannot tell that any of this happened. This is the single most useful power in a harness, and most beginners never use it because they think of the transcript as a record rather than as working memory they curate.

Consequence two: the transcript is the debug log

If you cannot print the exact array sent at step seven, you cannot debug the agent — you are guessing about a program whose input you never looked at. Log it. Every step. Redact secrets, keep the rest.

Almost every "the model is being stupid" bug turns out, on inspection, to be a tool result that was empty, truncated, or an error string nobody read.

EXAMINATION

Agents and Automation — course exam

This paper covers the whole course:

- choosing between a script, a workflow and an agent
- the transcript, the tools and the loop
- the data layer underneath
- failure modes, traces, budgets and validators
- permissions, injection and blast radius
- the eight patterns that keep working
- shipping and operating
- the year after it ships

56 questions, the whole pool. The site draws 25 for a sitting, pass mark 70%. Work any 25 under your own conditions. Answers from page 432.

1 Module 1 · Before you build: script, workflow, or agent

A crontab line reads 0 6 13 * 5. When does the job actually run?

A. On the thirteenth of every month and on every Friday, because cron treats the two day fields as OR

B. Only on Friday the thirteenth, because both day fields must match before the job is allowed to fire

C. On the thirteenth of every month only, because day-of-month takes precedence over day-of-week in the five-field syntax

D. On every Friday only, because the day-of-week field is evaluated last and overrides whatever the day-of-month field says

2 Module 1 · Before you build: script, workflow, or agent

A task takes fifteen minutes a week, the build takes eight hours, and the integration breaks three or four times a year at about an hour each. Why might it still be worth doing?

A. Because the maintenance hours can be charged to a different budget line from the build itself

B. Because a payback period of seven months is well inside the normal tolerance for internal tooling

C. Because it gets done consistently, and at all, at six on the Monday after a long weekend

D. Because a model will improve over time and the same automation will handle progressively more of the work each year

3 Module 1 · Before you build: script, workflow, or agent

A team says an integration is 'just an API call'. What share of the work is the call, and what is the rest?

A. About half of it; the rest is error handling and the tests written around the call itself

B. About five per cent; the rest is auth, rate limits, pagination, retries, error mapping and secret storage

C. About ninety per cent, since modern client libraries handle pagination, retries and refresh tokens automatically for you

D. About half, because the remaining effort is entirely in mapping their data model onto your own schema

4 Module 1 · Before you build: script, workflow, or agent

Your stated control is that a human checks the output. Which single measurement tells you whether that control works?

A. The proportion of outputs the reviewer marks wrong, tracked weekly against the model's own error rate

B. The reviewer's agreement with a second reviewer on a shared sample, expressed as a percentage of items agreed

C. The average time the reviewer spends per item, since attention correlates closely with time spent on each one

D. The number of known-bad items you deliberately seeded that come back flagged

5 Module 1 · Before you build: script, workflow, or agent

A triage bot is given an 'I am not sure, send this to a person' output. What must be tracked alongside the error rate?

A. The share of abstentions a person subsequently agrees with, because that is the only measure of whether the exit was used correctly

B. The latency of the abstain path, since escalated items wait longer and that difference shows up in cycle time

C. The abstain rate, because at forty per cent you have added a queue rather than automated anything

D. The cost of the abstain path, because an escalation consumes a second model call as well as the human minutes it triggers

6 Module 1 · Before you build: script, workflow, or agent

The platform that already holds your data says the feature you were about to build is coming soon. What makes waiting a decision rather than a hope?

A. A dated commitment, or a beta you can log into today

B. A public roadmap page, because vendors are held to what they publish on it

C. An account manager's verbal assurance, provided it is confirmed in an email you keep on file for later

D. The fact that the feature exists in a competitor's product, since parity releases usually follow within a quarter or two

Module 1 · Before you build: script, workflow, or agent

The module starts on p. 11.

MODULE 1 TEST

Module 1 test, Q1, p. 50
An n8n flow classifies each incoming WhatsApp message with a model call, then branches to one of three fixed paths. Why is this still a…

Answer B: Because every path it can take was drawn by a person before it ran; the model fills one blank on a form

Why: The test is who decides what happens next. Here the author decided, in advance: three branches, drawn before the flow ever ran. A model inside one box makes the content uncertain and leaves the structure fixed. The tell of an agent is that you do not know how many steps a run will take before it starts.

Module 1 test, Q2, p. 50
A nightly job reads three input files. A supplier renames one, so that file stops arriving. Which line turns this into six weeks of a…

Answer A: total = row.get('total', 0), because a missing field becomes a plausible number and nothing raises

Why: Defaulting a missing field to a plausible value is how a deterministic pipeline fails silently. A crash is a message; a zero is a lie. The habit that catches the rest of the class is one sanity assertion at the end of the transform — this week's total within a sane multiple of last week's — because no exception is ever thrown by a well-formed wrong number.

Module 1 test, Q3, p. 51
A process takes 800 minutes a week. Automation removes 2.5 of the 4 minutes per case, 30 per cent of cases still need a phone…

Answer C: About 275 minutes, because only the automatable share of the clean cases counts and review time is subtracted

Why: A hundred and forty clean cases at two and a half minutes is 350 minutes; the review the automation creates adds 75 back. The saving is around 275, which is still a good result and a third of what the project would otherwise be sold as. The honest figure matters because it is what the project gets judged against later.

Module 1 test, Q4, p. 51
Sorting work by volume and by how much the inputs vary, which quadrant is the classic failed automation project?

Answer D: Low volume, high variance — runs four times a year and looks different each time

Why: Low volume with high variance is work a person should do. The build cost never comes back because the volume is small, and each instance differs enough that maintenance is continuous. Most failed automation projects are this quadrant with a budget attached. Low volume and low variance wants a checklist, not a build.

Module 1 test, Q5, p. 51

A ten-step agent sends about 60,500 input tokens per run. The same agent doing the same job in five steps sends about 19,000. Why is…

Answer C: The transcript is re-sent every step, so the growing part of the cost scales with the square of the step count

Why: The fixed prefix is paid once per step, and everything accumulated so far is paid again on every later step, so the growing part is proportional to $n(n-1)/2$. That is why removing five steps beats any amount of prompt tuning, and why a run twice as long costs roughly four times as much.

Module 1 test, Q6, p. 52

A team is choosing between a hosted visual tool and a Python file. Which fact about the pricing should shape the design before anything is…

Answer B: Both bill per task or operation, so a twelve-step flow run a thousand times bills twelve thousand units

Why: A task or an operation is roughly one step doing one thing, and a loop over fifty rows is fifty of them. Teams routinely design an elegant twelve-step flow and then find it costs more than the person it replaced. The design response is fewer, fatter steps, and pushing loops into a single call to something you wrote.

THE LESSON CHECKS

Lesson 1, p. 15

An n8n flow receives a support email, uses a model to sort it into billing, technical or other, routes it into one of three fixed…

Answer A: No — every path it can take was drawn before it ran; the model only picks between branches somebody else authored.

Why: An agent is a program where the model, not the author, decides what happens next.

B Yes — a language model is…: Treats any model-made choice as agency, missing that the full set of possible paths was fixed at build time.

C Yes — it takes real actions…: Equates autonomy from humans with agency; by that test an unattended cron job would also be an agent.

D No — it uses the model…: Believes the function-calling interface is what makes an agent, rather than who controls the loop.

Lesson 2, p. 19

A nightly job pulls yesterday's orders and emails a revenue total. The upstream team renames the `total` column to `total_amount`. The job keeps running, and…

Answer A: Read the field with `row['total']` instead of `row.get('total', 0)`, so a missing column raises instead of quietly producing a plausible number.

Why: Most automation requests are a scheduler and thirty lines of code; put a model only in the one step where the input is genuinely unstructured.

B Add a model to the pipeline…: Reaches for intelligence where the problem is a silent default; the model cannot know a value is missing either, because nothing failed.

C Move the job from cron to…: Retries a job that is succeeding. Nothing errored on any of the nine nights, which is exactly the problem.

D Run the job hourly instead of…: Produces the wrong answer more often rather than making a wrong answer detectable.

Lesson 3, p. 22

A finance team spends about 800 minutes a week on invoice processing. Watching them shows that 60 per cent of the elapsed time is waiting…

Answer A: The automation has a ceiling of roughly 15 per cent, so the approval queue is the larger target and may not need a model at all.

Why: The process on paper is not the one that runs, and the ceiling on any automation is the share of elapsed time held by the steps it actually replaces.

B A faster model will shorten the…: Assumes the queue is limited by upstream throughput, when the delay is a person's attention and unchanged by anything arriving sooner.

C The 60 per cent is idle…: Counts waiting as work being replaced; the manager still has to read and decide, so that time does not vanish when the typing does.

D Data entry is the only mechanical…: Chooses the step that is easiest to automate rather than the one that holds the time, which is how projects deliver a tenth of what was promised.

BUILD WITH IT

Agents and Automation

What an agent actually is, and when you should build a script instead.

WHAT IT TEACHES	Past the hype: what an agent actually is, how the loop works, why most agent demos collapse in production, and how to build automation that survives contact with real data.
WHAT IS INSIDE	Eight modules and 72 lessons, 27 figures drawn from data, eight case studies, eight module tests, a 56-question examination, and every answer with the reason behind it.
LICENCE	CC BY-NC-SA 4.0. Copy, print, share and adapt it for any non-commercial purpose, with credit, under the same licence. There is no paid edition.
THE COURSE	Free to read, with the lessons, the films and the examination, at **addaly.com/learn/agents-and-automation**

Addaly

First edition, September 2026 · build 62a26075

Scan to open the course