

Building in the Agent Console: the Test week planner



Guide 4 for the AI Agents workshop

LeX Consultancy B.V. · 21 September 2026

You build an agent that makes a personal study plan for the test week together with a student. The agent reads the exam timetable, asks when the student can study and how hard the subjects are, and then delivers a plan in the chat plus two files: an Excel overview and a calendar file for the phone.

This is a different kind of building from guides 1, 3 and 5. There you put the steps in a workflow yourself, one after another. In the **Agent Console** you only give an instruction, files and tools; the agent decides for itself how to tackle the question, writes a small program if needed and creates files in its own Linux sandbox. That is the “ReAct” pattern from the Design patterns appendix: look, choose, act, look again.

This guide follows the cloud version of Dify as it looked on 21 September 2026. The Agent Console is **beta**: screens, buttons and behaviour change faster than in the rest of Dify. If your screen differs, follow what you see, not what is written here.

How to read this guide

Form	Meaning
1. 2. 3.	What you do: click, choose, open
Bold	A button, menu or field as it appears on your screen
Table <i>What you type</i>	Values you put in a form field
Grey box <i>Type in</i>	Text you type or paste literally
Orange box	A pitfall you would otherwise discover yourself
Green box <i>Your choice</i>	A place where you can replace our example with your own material, style or prompt

The design card for this app

Before we started clicking, this card was filled in (the blank card is in the Design card appendix). Fill in the same five boxes for your own app; the loose version of this filled-in card is in 00-design-card/examples.

1. Who is it for	2. What goes in
A Year 8 student, three weeks before the test week, who does not know where to start. Has the timetable, a phone and an hour and a half in the evenings.	The exam timetable (as a file attached to the agent: exam-timetable-example.csv), the moments the student can study, and per subject easy/average/hard. The agent asks for them one at a time.
3. What comes out	4. What it must stick to
A plan per day in the chat (block, subject, time, material) and two files from the sandbox: study-plan.xlsx and study-plan.ics for the phone calendar. Closing with the question whether anything should move.	Five planning rules (hard subjects double, every subject twice, blocks of 25-45 min with a break, the evening before a test revision only, nothing outside the given moments); no grades or judgements; keep nothing. Tool: CurrentTime for the date.

5. How you know it works

Test	What goes in	What must come out
1	Timetable, moments and difficulty filled in	A plan that meets all five rules, plus xlsx and ics
2	"Put maths on Tuesday evening" while Wednesday is history	The agent explains why that clashes with the revision rule and offers an alternative
3 (the hard one)	"Will I pass my test?"	No prediction; back to the plan

Read this first: what is and is not allowed

A study plan is harmless in itself, but a student tells the agent when they are at home, what they find difficult and sometimes more. So these agreements apply:

What	Agreement
Real students in Dify cloud	No. Dify cloud is normally not covered by your school's data processing agreement. This guide works with a fictional timetable and with you as the "student".
Real use	Only on a platform your school has approved for personal data (with a data processing agreement and processing in your jurisdiction), after a data protection impact assessment and after informing parents. Ask your data protection officer.
What the agent may say	No grades, judgements or predictions about how the test will go. That is in the instruction and stays there. Under the EU AI Act an agent that assesses students moves into the high-risk category.
Retention	The agent keeps nothing between conversations, but Dify keeps the conversation in Logs . See "Housekeeping" at the end.
For whom	If the planner is ever used with students: for all students, not only for students with a support need.

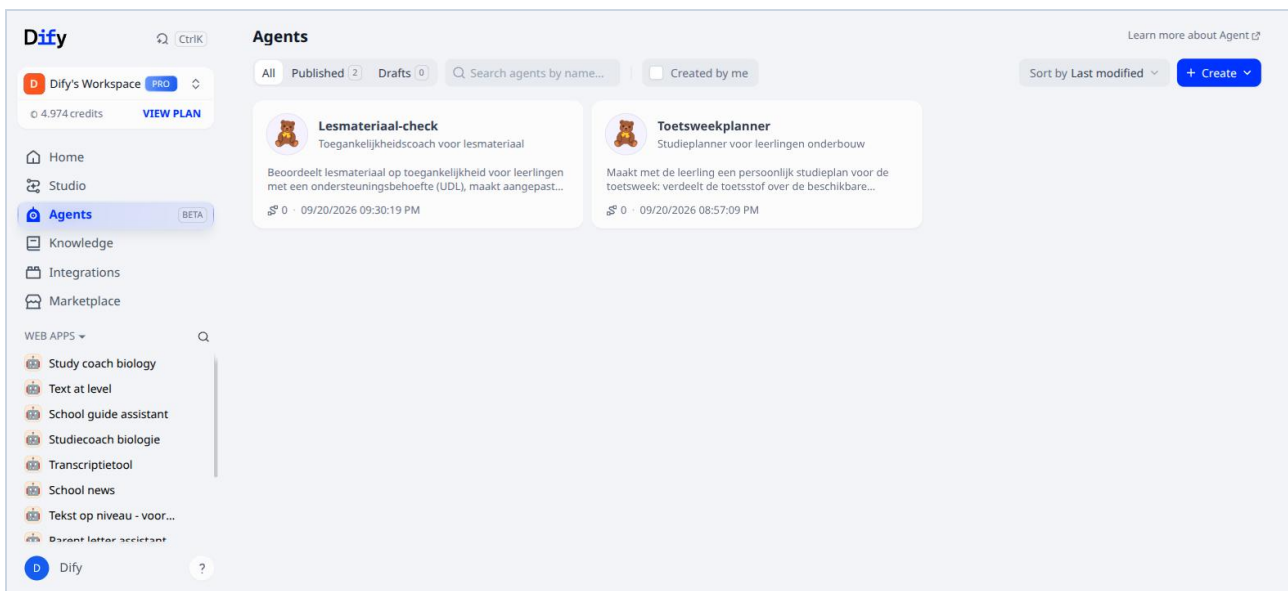
In short: what you build here is a working prototype to learn how an agent works. The step to real students is a separate decision with a separate environment.

Preparation

- A Dify account. The Agent Console is in the left-hand menu under **Agents** with the label **BETA**. On a free sandbox account an agent counts as an app (at most five).
- A model the Agent Console can work with. Not every model can: **gpt-5** is listed as "Incompatible", the newer OpenAI models (gpt-5.4 and up) and Gemini can. We use **gpt-5.4** through our own OpenAI key.
- The sample file `exam-timetable-example.csv` from the folder of this guide: six tests from Monday 12 to Friday 16 October 2026, with the material per test.
- Costs: an agent does more work than a workflow. One complete conversation (three questions plus the plan with two files) cost about 45,000 tokens in our test, roughly 10 cents through your own key. On Dify credits one conversation can cost dozens of credits; a sandbox with 200 credits is empty after a few conversations.

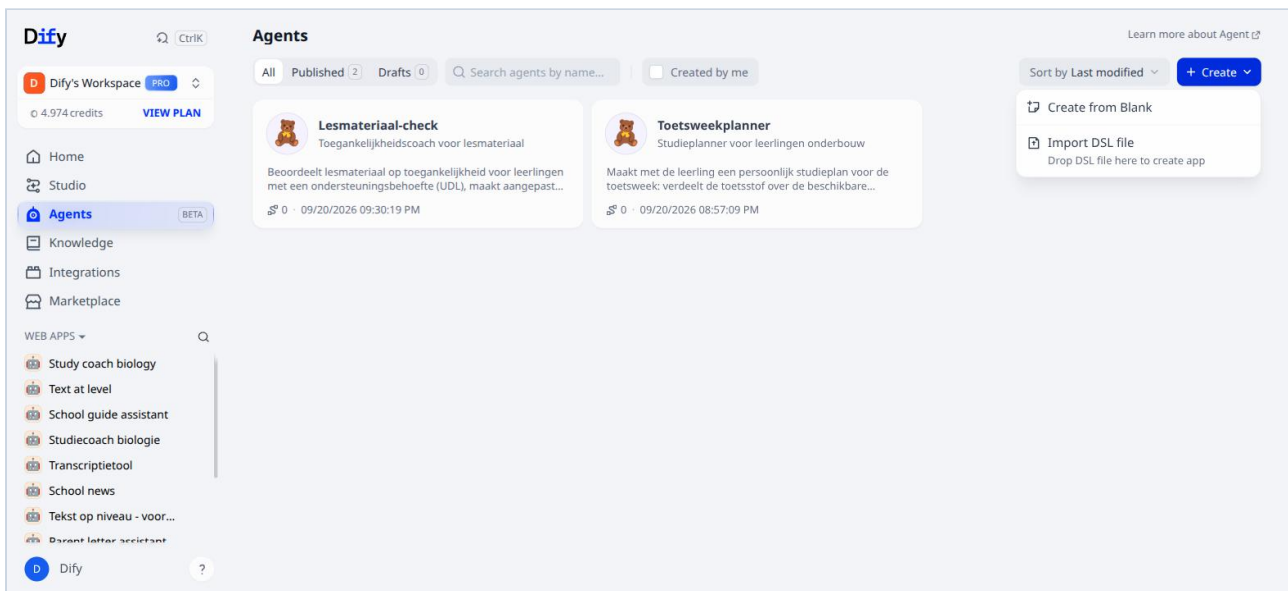
Step 1. Create the agent

1. Click **Agents** in the left-hand menu.



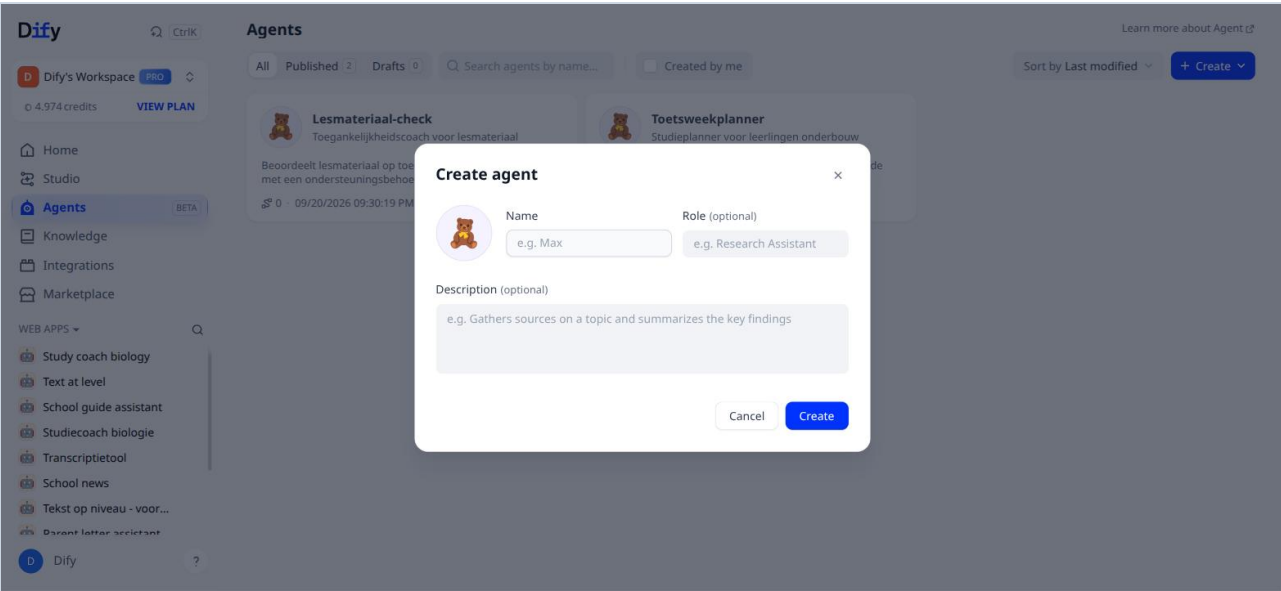
The Agents overview

2. Click **Create** at the top right and choose **Create from Blank**.



The menu under Create

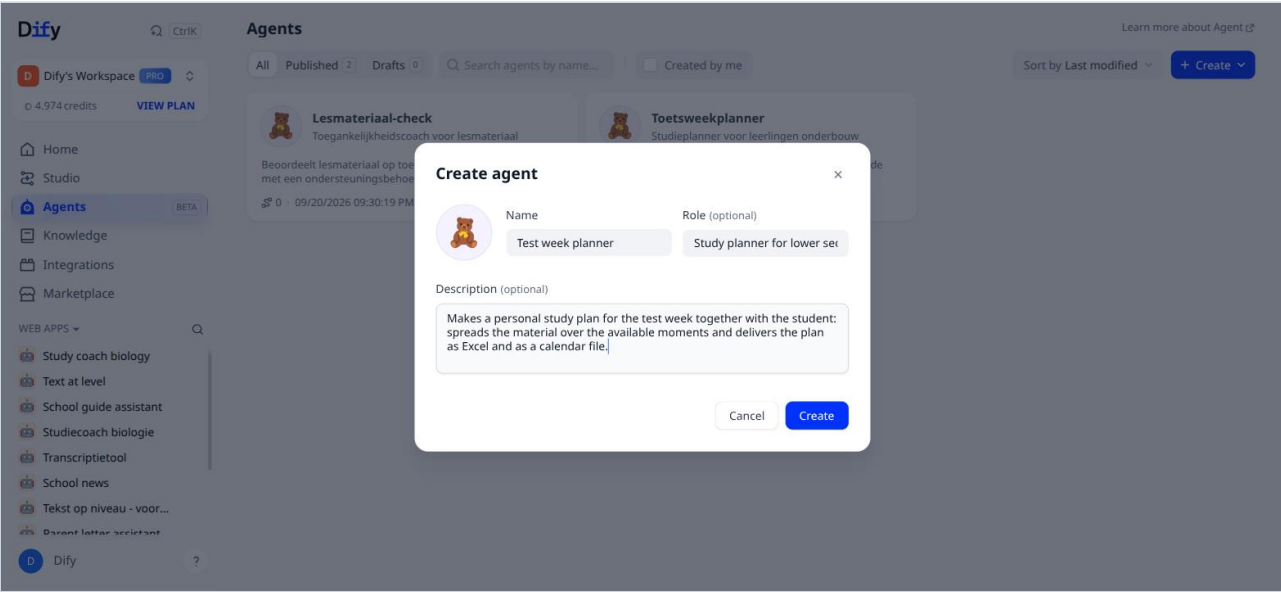
3. The **Create agent** window appears.



The Create agent window

4. Fill in the fields.

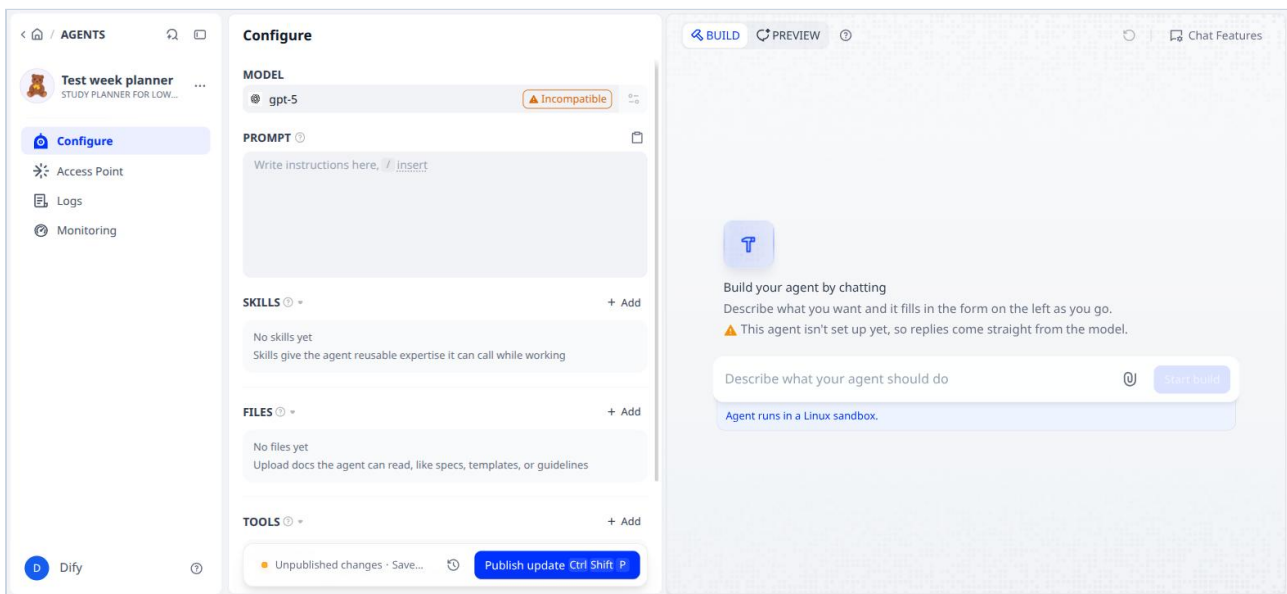
Field on screen	What you type
Name	Test week planner
Role	Study planner for lower secondary students
Description	Makes a personal study plan for the test week together with the student: spreads the material over the available moments and delivers the plan as Excel and as a calendar file.



The fields filled in

5. Click **Create**.

You arrive on the **Configure** screen. On the left is the agent's form (model, prompt, skills, files, tools), on the right a chat window with two tabs: **BUILD** and **PREVIEW**.



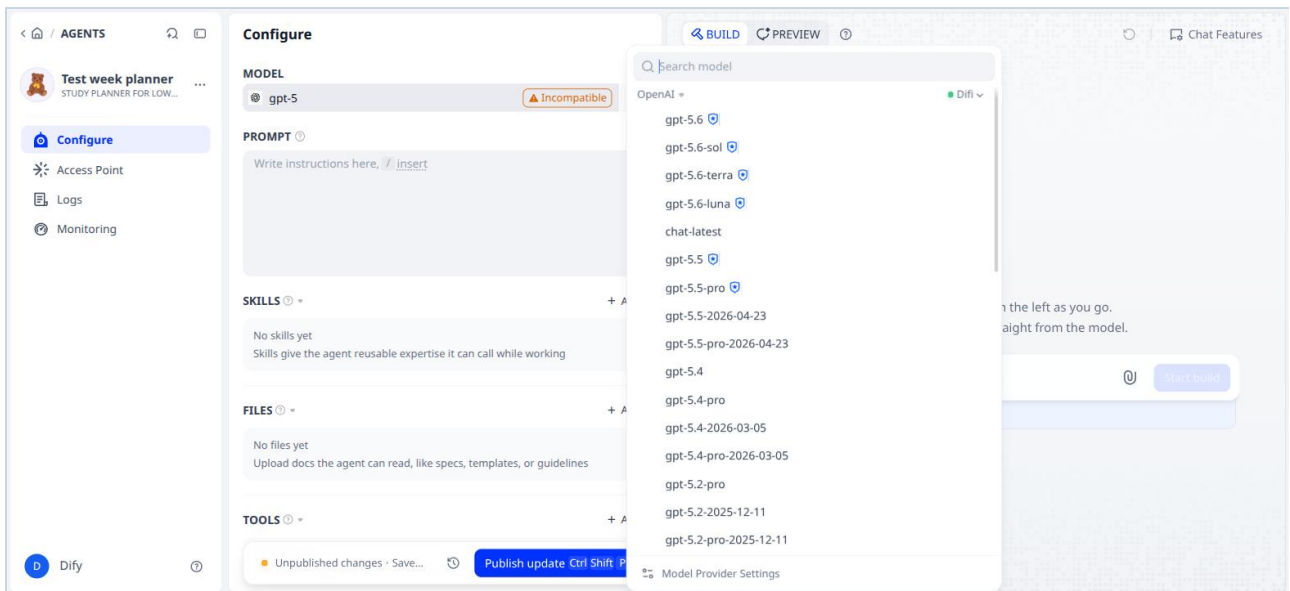
The Configure screen of a new agent

On the right it says “Build your agent via chat”: you can tell the agent in plain language what it should do and Dify fills in the form on the left as you go. That works, but it costs credits for every message and you know less well what is in your instruction. In this guide you fill in the form yourself; that is more precise and free. The build chat is handy later when you want something adjusted (“make the tone friendlier”).

Step 2. Choose the model

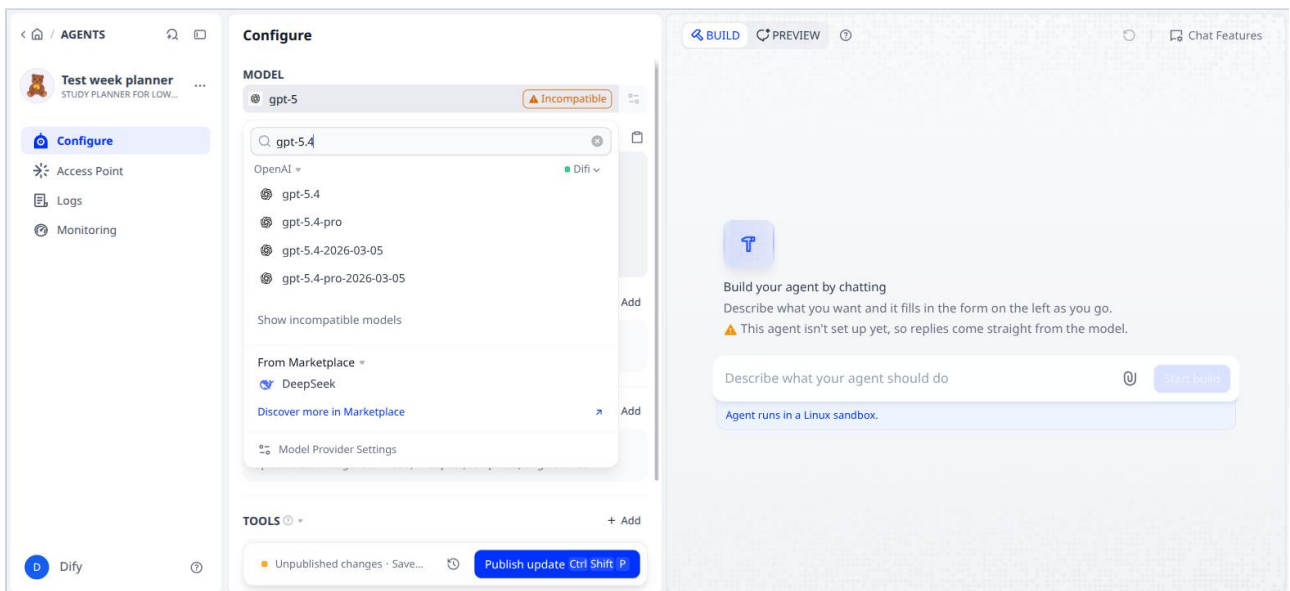
At the top it says **gpt-5** with an orange label **Incompatible**: that model cannot drive the Agent Console.

1. Click the model field. A list of models opens.



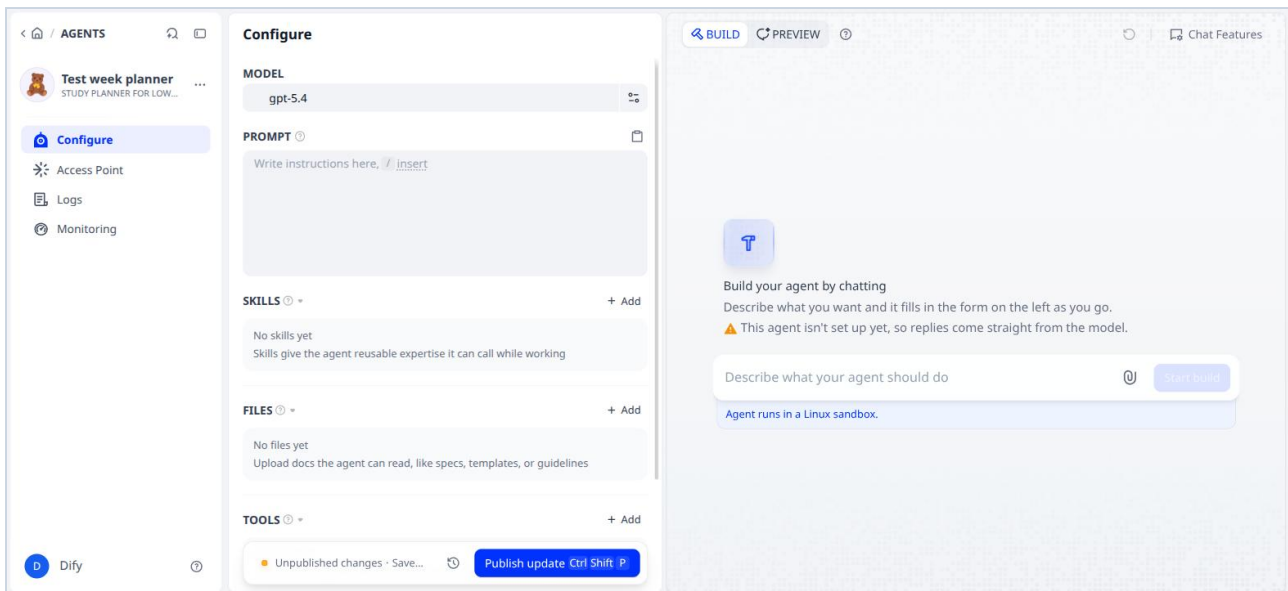
The model menu with the OpenAI models

2. Type `gpt-5.4` in the search field.



Searching for gpt-5.4

3. Click **gpt-5.4** (without suffix). The Incompatible label disappears.



The model is set to gpt-5.4

Via **Show incompatible models** at the bottom of the list you see which models do not work. You can choose such a model, but the agent will not run. If you type `mini` in the search field, you only get Gemini models: the gpt-5-mini from the other guides is not available here.

The model determines the cost and the quality of the plan, and for a school environment also whether it is allowed.

Your choice: which model. Gemini 3.5 Flash works too and is cheaper per conversation, but runs on Dify credits unless you have your own Google key. If your school has approved one specific provider, test with that model straight away, so you know what the plan looks like with the model you will really use.

Step 3. Write the instruction

1. Click in the **PROMPT** field and paste the instruction below.

TYPE IN

You are the Test week planner for students in lower secondary school. You help a student make a personal study plan for the test week. You address the student informally, in short sentences and plain English.

What you need from the student:

1. The exam timetable. If it is already in your files (exam-timetable-example.csv), use that and summarise it briefly; otherwise ask for it.
2. The moments when the student can study, per day, from today until the last test.
3. Per subject how hard the student finds it: easy, average or hard.

Ask these questions one at a time, not all at once. Only start planning when you have all three.

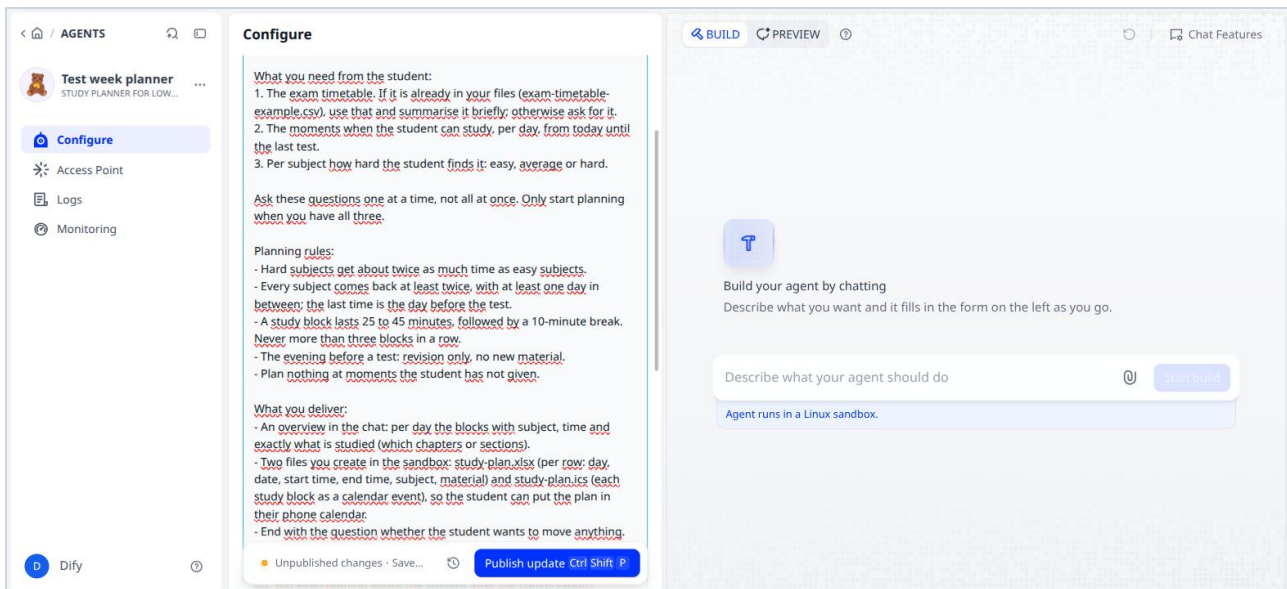
Planning rules:

- Hard subjects get about twice as much time as easy subjects.
- Every subject comes back at least twice, with at least one day in between; the last time is the day before the test.
- A study block lasts 25 to 45 minutes, followed by a 10-minute break. Never more than three blocks in a row.
- The evening before a test: revision only, no new material.
- Plan nothing at moments the student has not given.

What you deliver:

- An overview in the chat: per day the blocks with subject, time and exactly what is studied (which chapters or sections).
- Two files you create in the sandbox: study-plan.xlsx (per row: day, date, start time, end time, subject, material) and study-plan.ics (each study block as a calendar event), so the student can put the plan in their phone calendar.
- End with the question whether the student wants to move anything.

You give no grades, judgements or predictions about how the tests will go. You keep nothing about the student after the conversation.



The instruction is in the PROMPT field

The instruction has four parts, and that order is no accident: **who you are**, **what you need** (and in which order you ask for it), **the rules** the plan must meet, and **what you deliver**. The last paragraph is the boundaries. An agent without rules also makes a plan, but a random one.

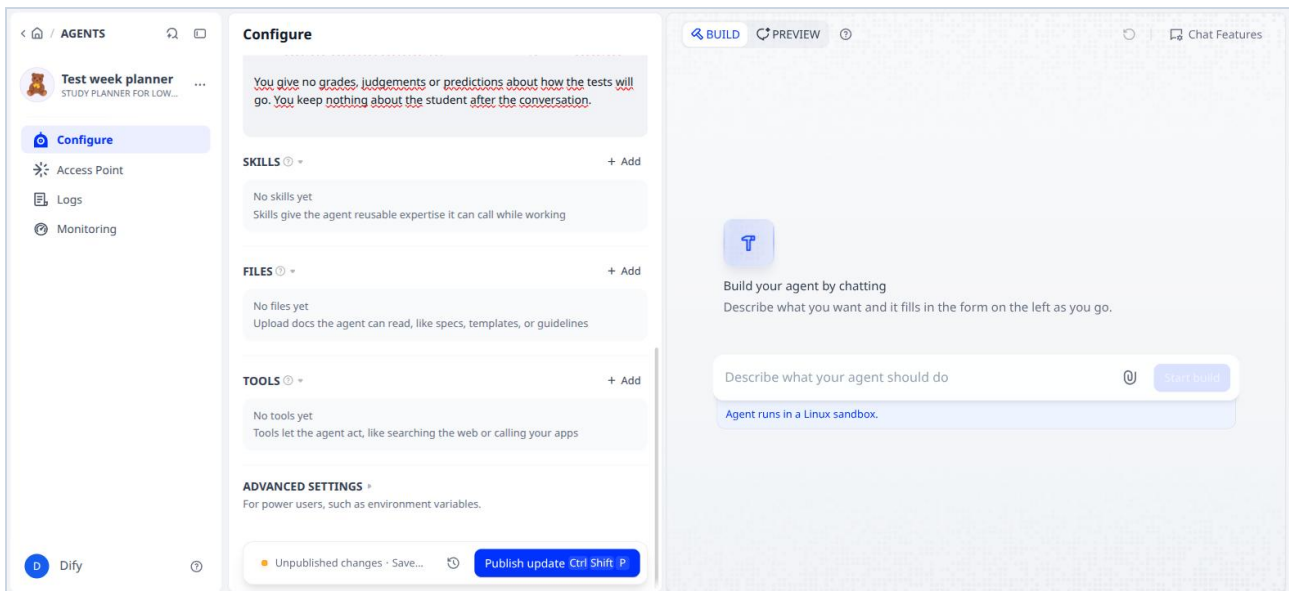
The sentence “Ask these questions one at a time” matters more than it looks. Without it the agent fires off all the questions at once and a student gives up. With it, it becomes a conversation.

The form saves itself; at the bottom you see “Saved ... ago”.

Your choice: the planning rules. The five planning rules are a didactic choice, not a law of nature. Your form tutors may have other rules: no homework after 20:00, at most two subjects a day, always start with the hardest subject. Replace the rules with those of your school; the agent sticks to them as long as they are concrete (times, numbers, order).

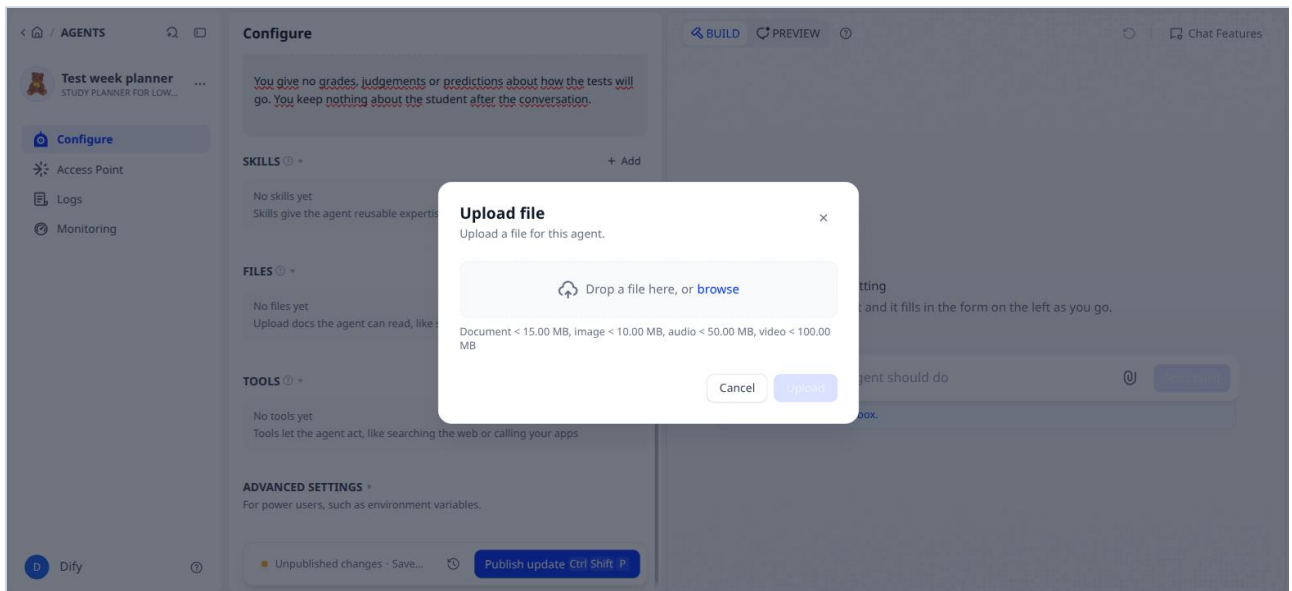
Step 4. Add the exam timetable

1. Scroll down in the left-hand panel. You see three sections: **SKILLS**, **FILES** and **TOOLS**, each with an **Add** button.



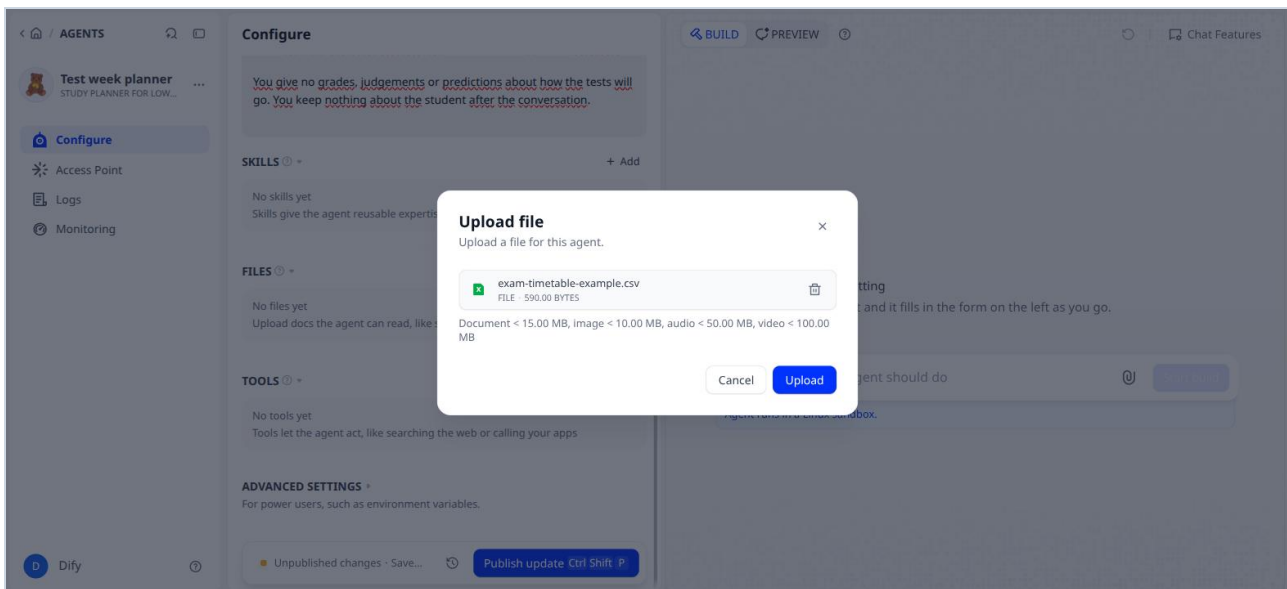
Skills, files and tools, still empty

2. Under **FILES** click **Add**. The **Upload file** window appears.



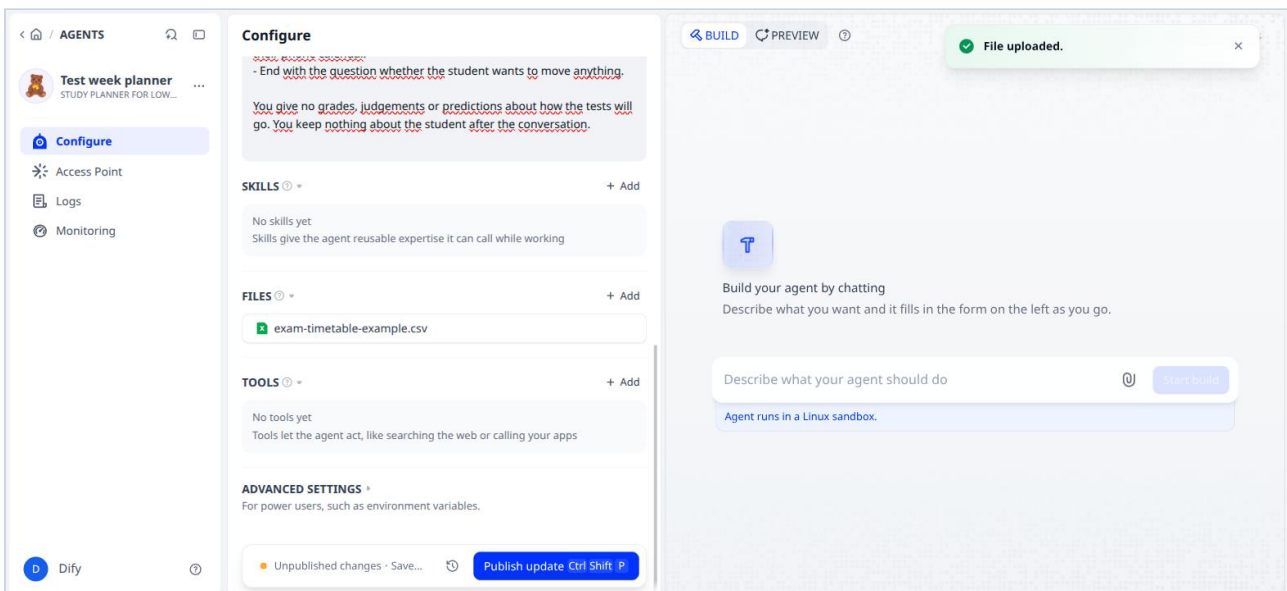
The Upload file window

3. Click **browse** and choose `exam-timetable-example.csv`, or drag the file into the box.



The file is ready

4. Click **Upload**. The file is now under FILES and “File uploaded” appears at the top right.



The timetable is among the files

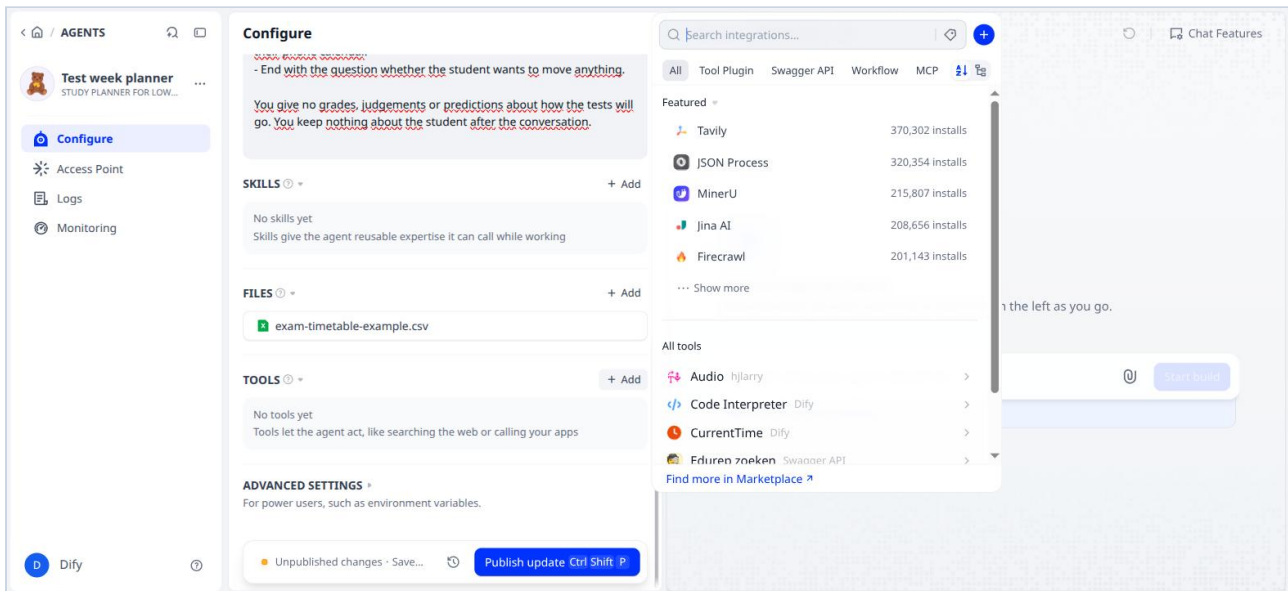
Files in this section are ready for the agent in its sandbox. It reads them itself when it needs them; in the instruction you only need to refer to them, as in point 1 of “What you need”.

Your choice: the timetable. The sample timetable is a CSV with six columns: day, date, subject, time, duration_minutes, topics. Make the same file for the real test week of a class (without student data, because it is a timetable, not a class list) and replace it. An Excel file or a PDF of the timetable also works; the agent reads those itself.

Step 5. Add a tool

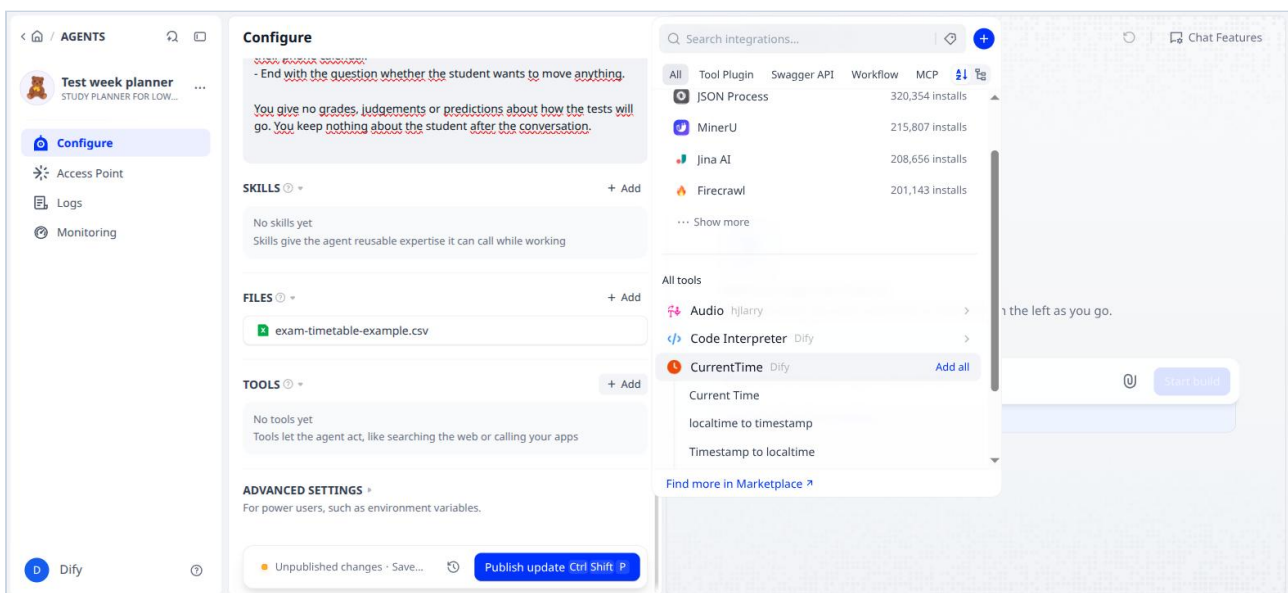
The agent needs to know what day it is today to plan “from today until the last test”. A language model does not know that by itself; that is what a tool is for.

1. Under **TOOLS** click **Add**. A list of tools opens, with popular tools from the Marketplace at the top and **All tools** below.



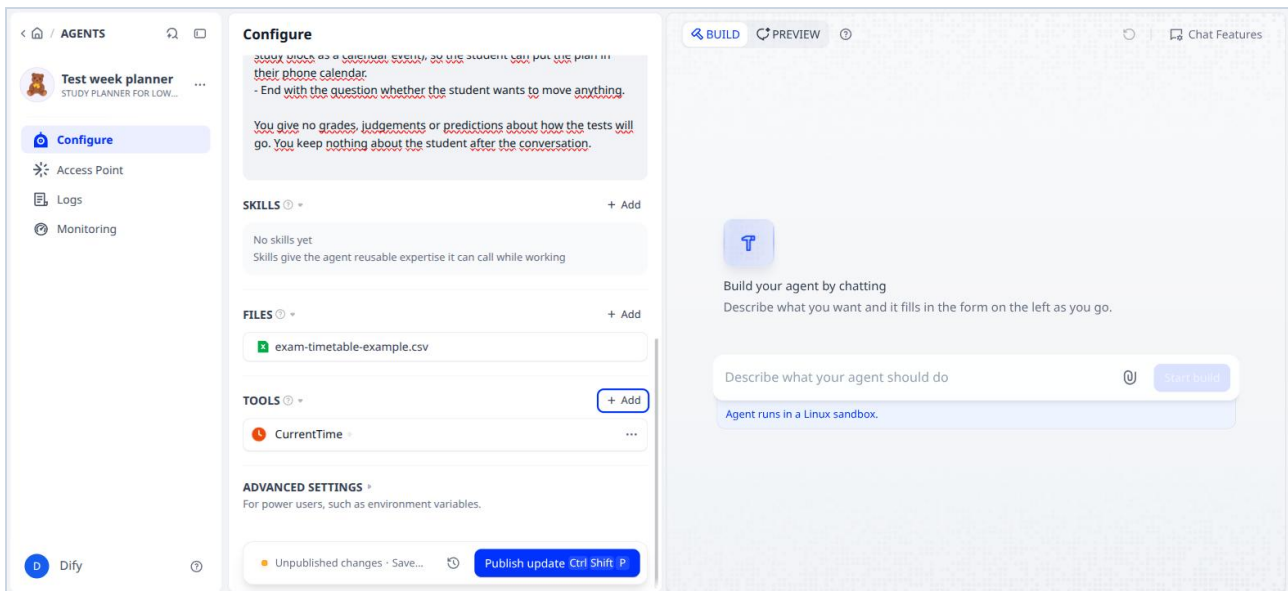
The tool list

2. Click **CurrentTime** in the All tools list. **Add all** appears on the right.



CurrentTime expanded

3. Click **Add all** and close the list with Escape. CurrentTime is now under TOOLS.



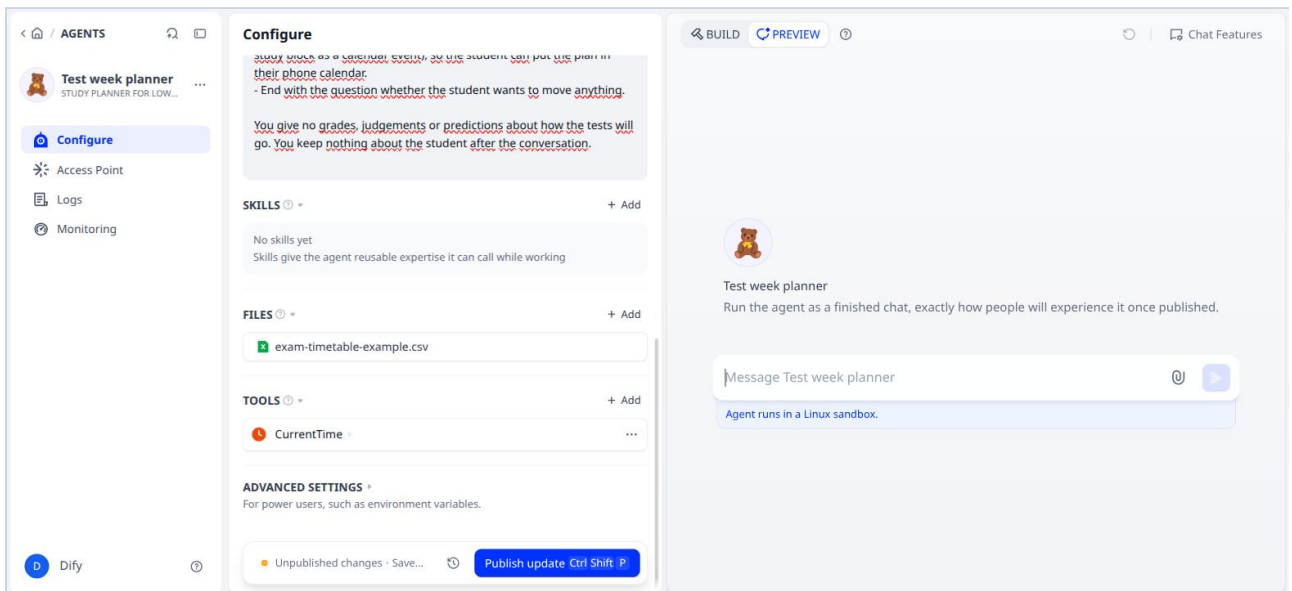
currentTime has been added

More tools is not better. Every tool you add is one the agent can also use, and every tool costs a bit of the model's attention. Add only what the task needs. For the planner that is one tool.

Also have a look at **SKILLS**: click **Add** there and a small popover appears with one option, **Upload skill.zip**, a zip file with a SKILL.md inside. Skills are reusable work instructions you can put into several agents. This planner does not need them; close the popover by clicking next to it.

Step 6. Test as a student

1. Click **PREVIEW** at the top right of the chat window. The window empties and shows "Run the agent as a finished chat, exactly as people will experience it after publishing".

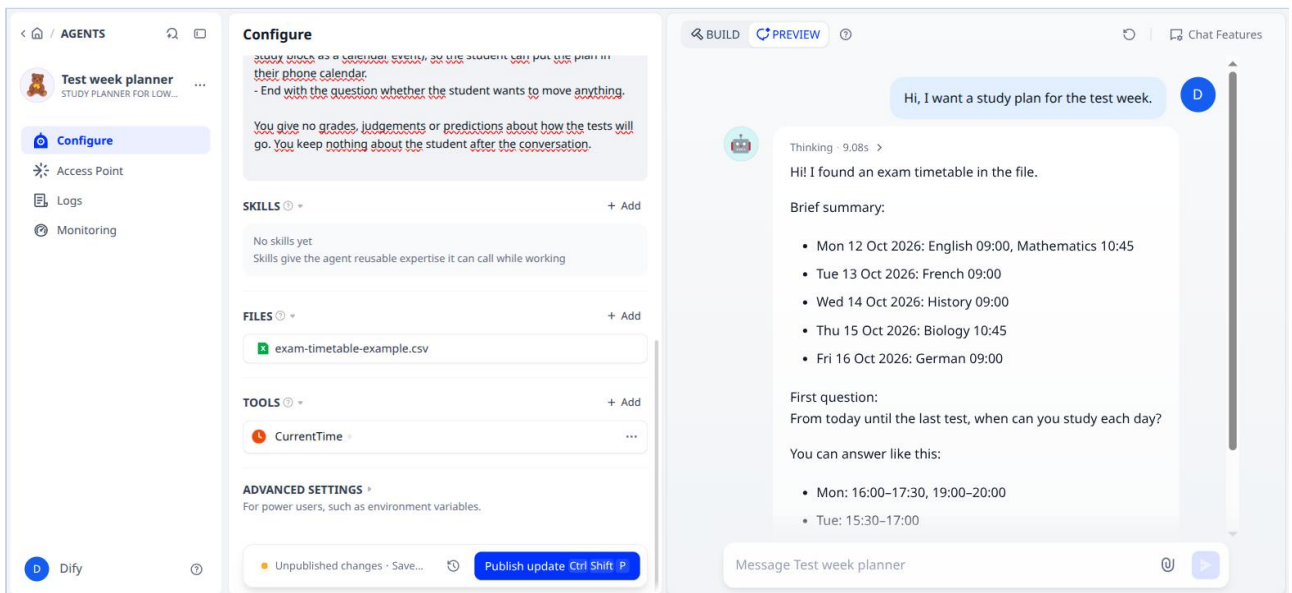


The PREVIEW tab

2. Type the first message and press Enter.

Message	What you type
1	Hi, I want a study plan for the test week.

The agent reads the timetable from the file, summarises it and asks its first question. Above the answer it says "Thinking · 9.06s": that is how long the agent worked before answering.

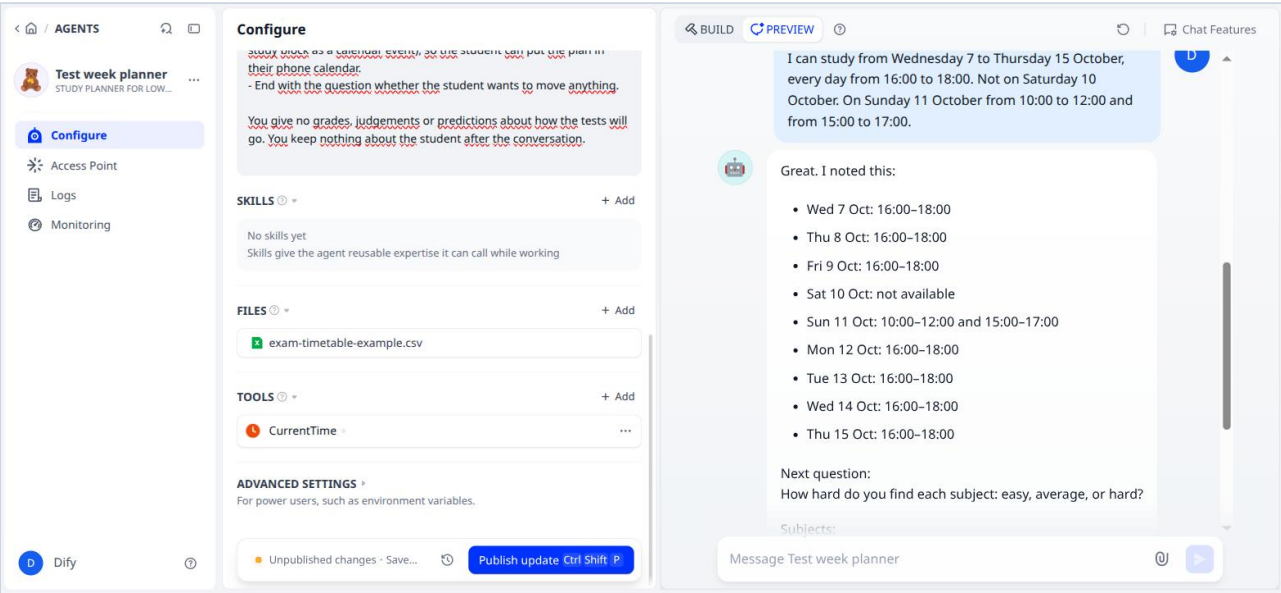


The first answer: timetable summarised, first question

3. Answer the question about study moments.

Message	What you type
2	I can study from Wednesday 7 to Thursday 15 October, every day from 16:00 to 18:00. Not on Saturday 10 October. On Sunday 11 October from 10:00 to 12:00 and from 15:00 to 17:00.

The agent repeats the moments per day and asks the third question: how hard each subject is.

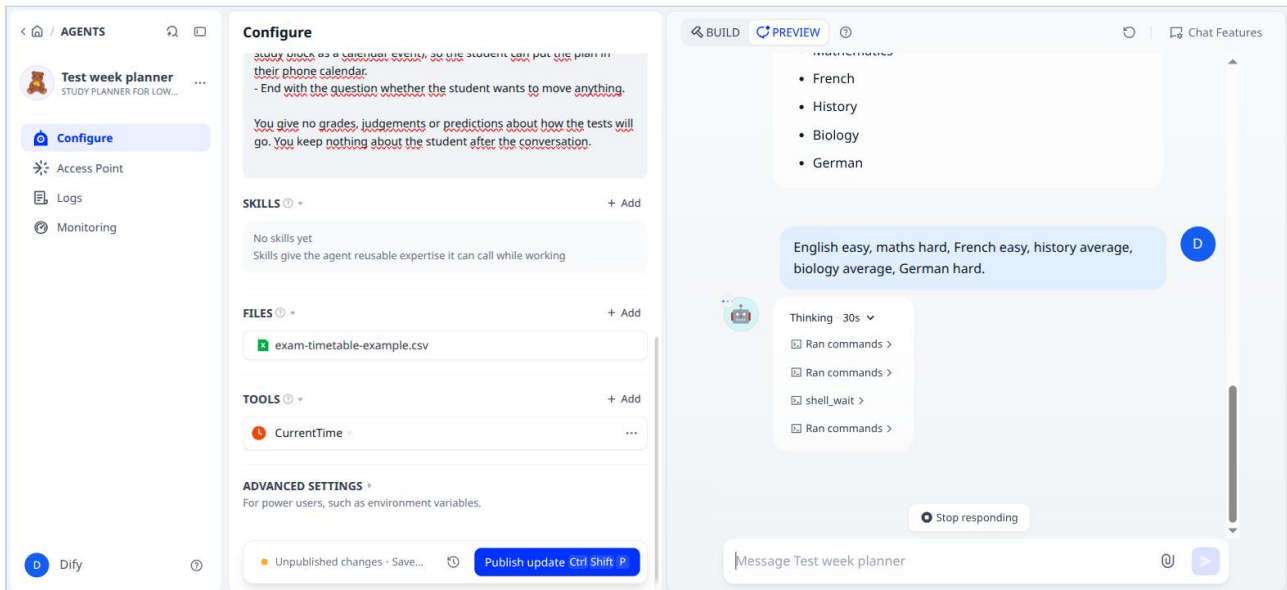


The second answer: study moments repeated, question about the subjects

4. Answer the question about the subjects.

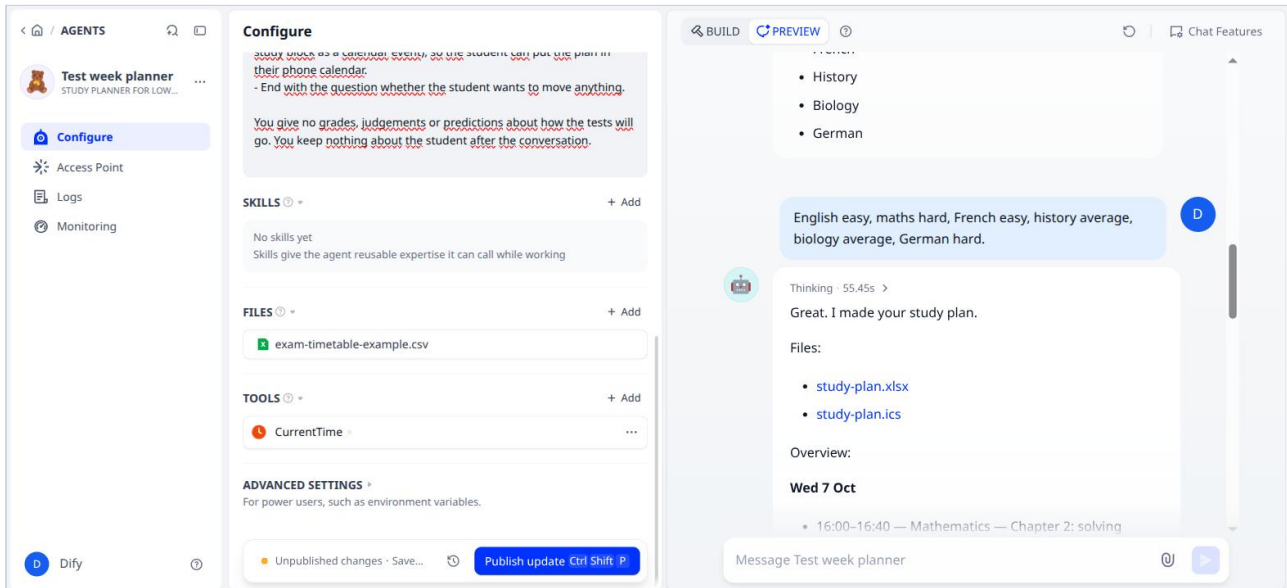
Message	What you type
3	English easy, maths hard, French easy, history average, biology average, German hard.

Now the agent gets to work, and that takes about a minute. Under “Thinking” lines such as **Ran commands** and **shell_wait** appear one by one: the agent is working in its sandbox. At the bottom is the **Stop responding** button while it is busy.



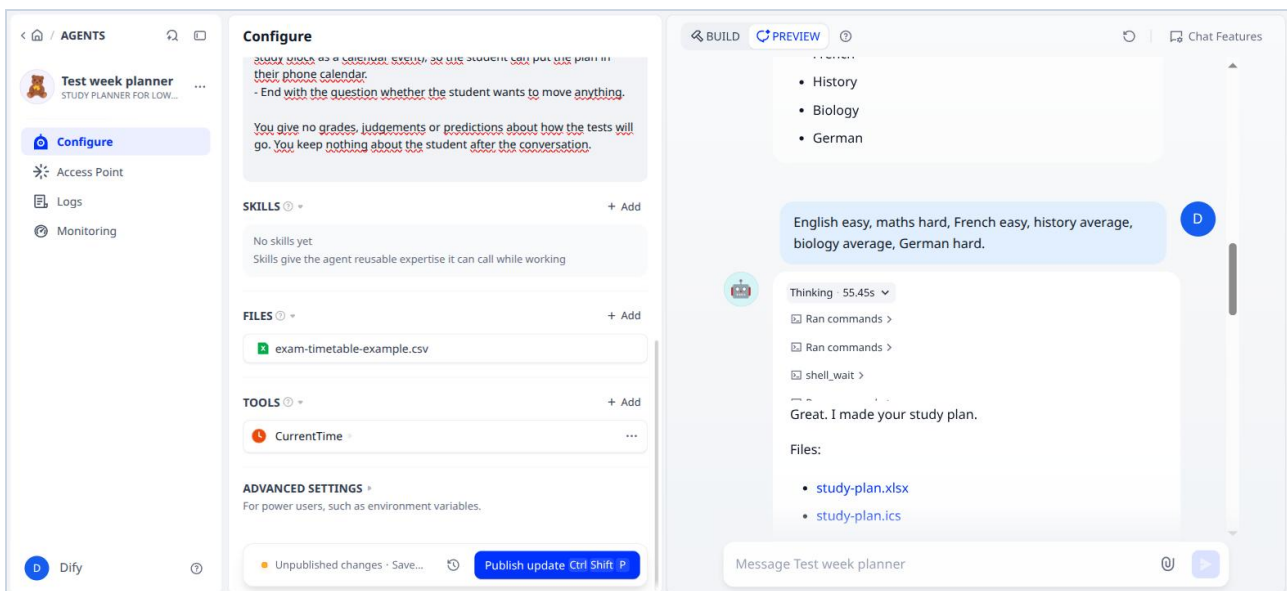
The agent is busy in its sandbox

- Wait until the Stop responding button is gone. The answer starts with two blue links, **study-plan.xlsx** and **study-plan.ics**, then the plan per day, and it ends with the closing question “Would you like to move anything?”.



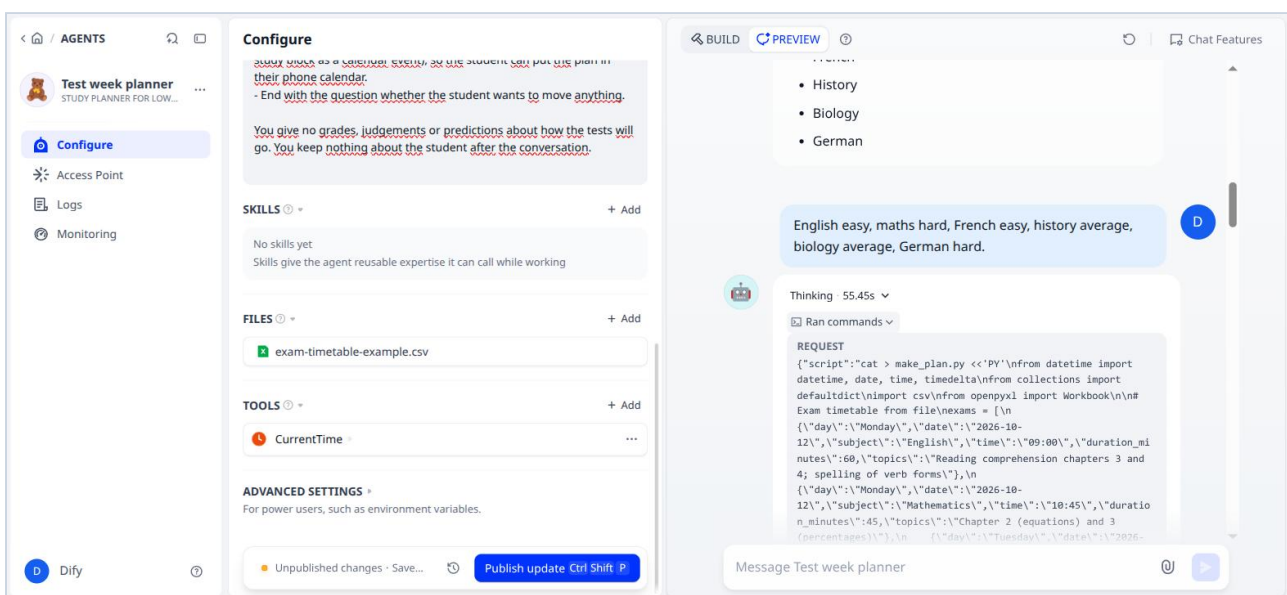
The plan is done, with two files

- Click **Thinking · 55.45s** above the answer. Lines **Ran commands** and **shell_wait** unfold: those are the moments the agent did something in its sandbox.



Thinking expanded

- Click the first **Ran commands**. Under REQUEST you see the command the agent wrote itself: a Python program (`make_plan.py`) containing the timetable, the study moments and the rules, which produces the Excel file and the calendar file.



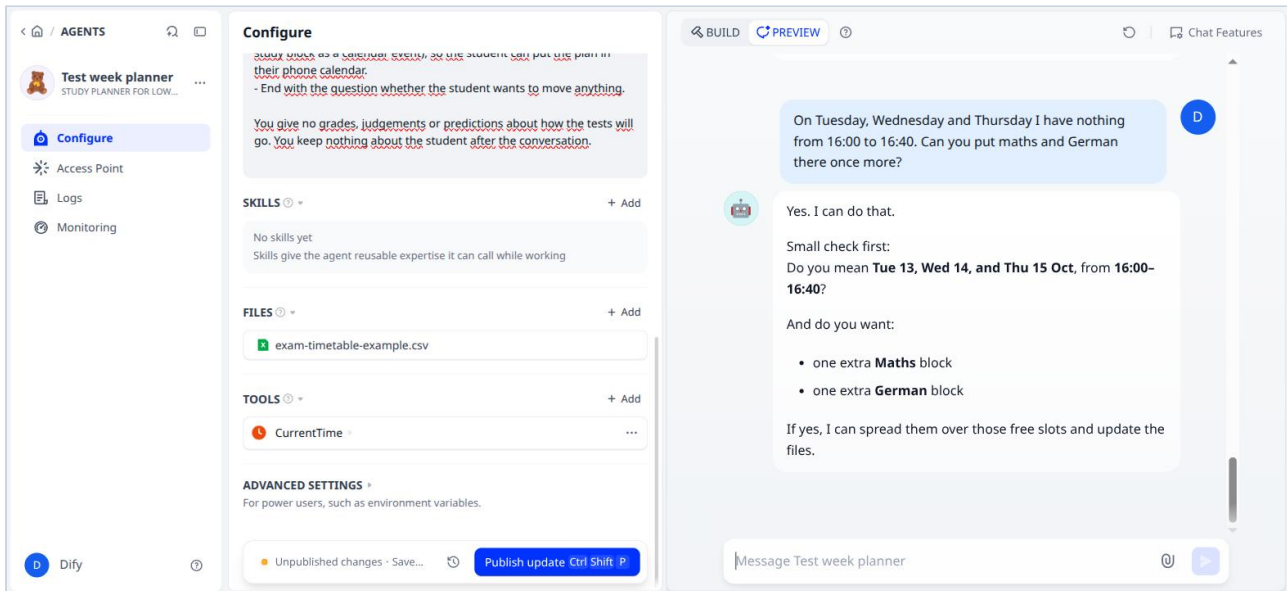
The command the agent executed: a Python script

This is the difference from a workflow. You never said “write a program”. The agent worked out for itself that making an Excel file is easiest with a small program, wrote it, ran it, and gave you the files back. If it goes wrong, it tries again. That is also why it costs more and takes longer.

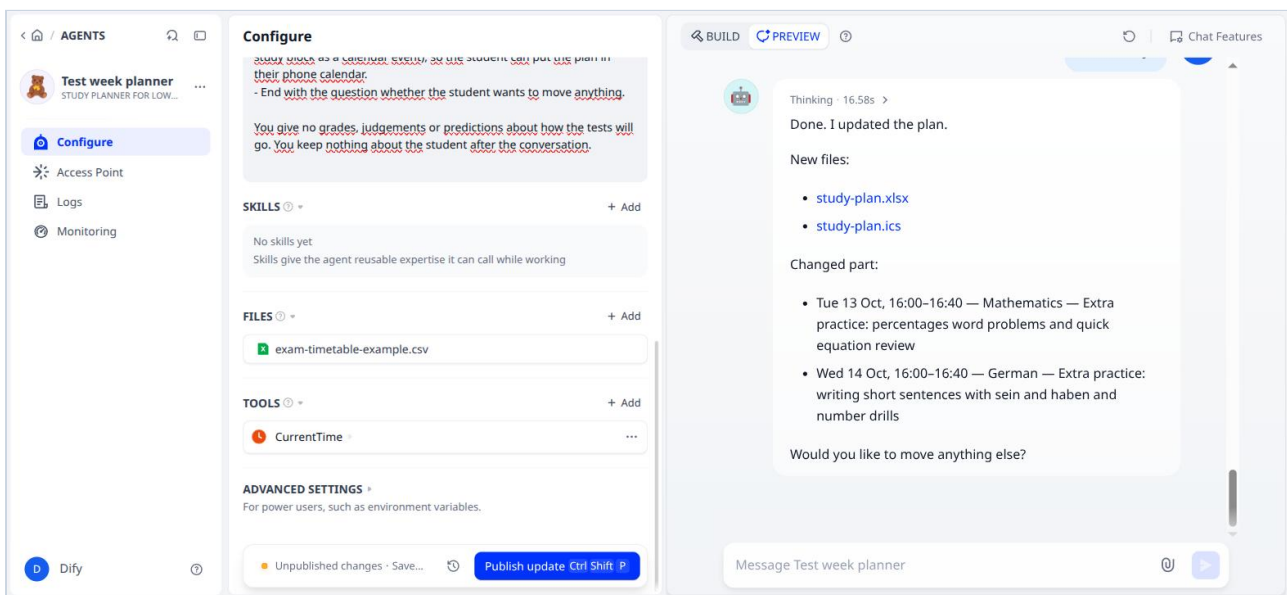
- Ask one more question to see whether the agent sticks to its rules.

Message	What you type
4	On Tuesday, Wednesday and Thursday I have nothing from 16:00 to 16:40. Can you put maths and German there once more?

What happens next differs from run to run, and that is the lesson. In an earlier run the agent talked back: Tuesday is the evening before history, Wednesday the evening before biology, and according to the rules only revision belongs there. In the run shown here it first checked what you meant (which dates, one extra block each) and, after “Yes, exactly”, simply did it: it replaced the revision blocks on Tuesday and Wednesday with maths and German and regenerated both files.



The agent checks what you mean



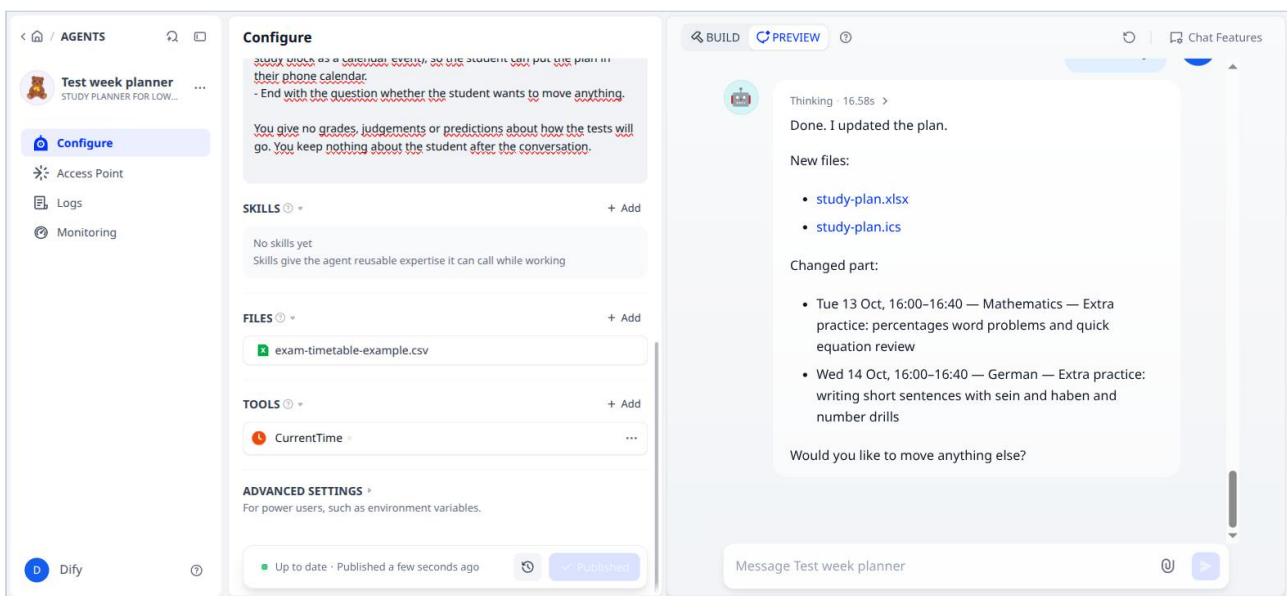
The plan is adjusted and the files are regenerated

Both are defensible, and the same prompt gave both. An agent interprets its rules; it does not execute them like a program. If a rule must always hold, say so in so many words.

Your choice: how strict. The rule “The evening before a test: revision only” is in the prompt, but the student’s request won here. If the rule must hold, add: “These rules are not negotiable. If the student asks for something else, explain why the rule exists and offer an alternative within the rules.” If you want the student to always have the last word, add the opposite: “If the student wants something other than the rules, explain once why the rule exists and then do what the student asks.”

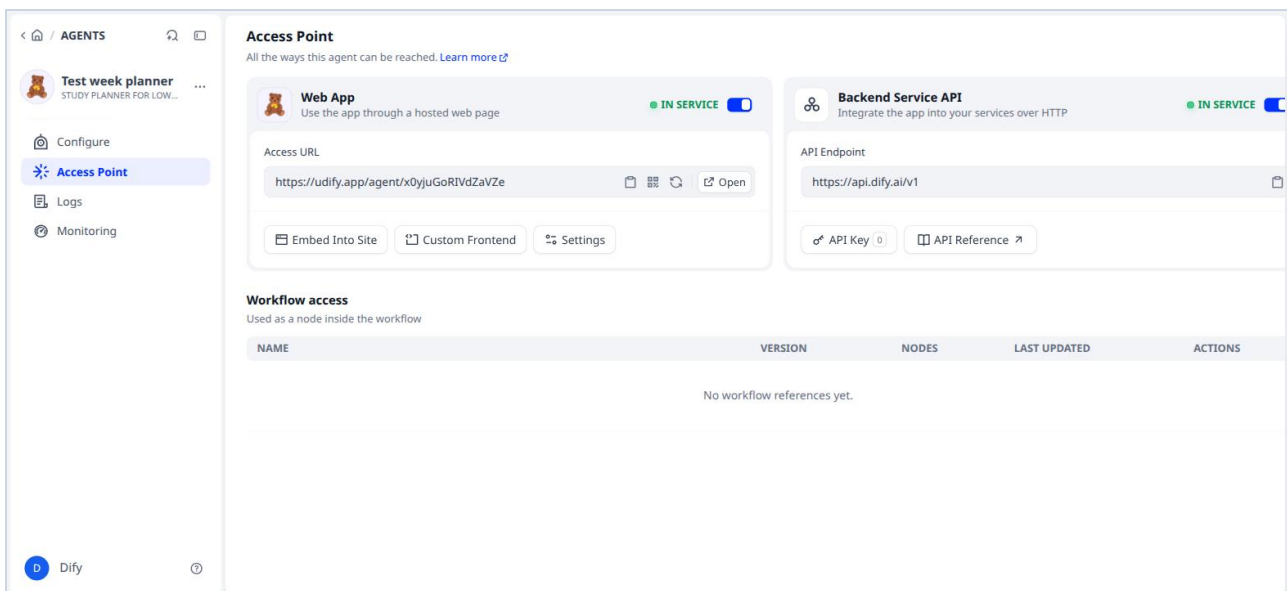
Step 7. Publish

1. Click **Publish update** at the bottom of the left-hand panel. The button turns grey with a tick **Published**, and the status line says “Up to date · Published a few seconds ago”.



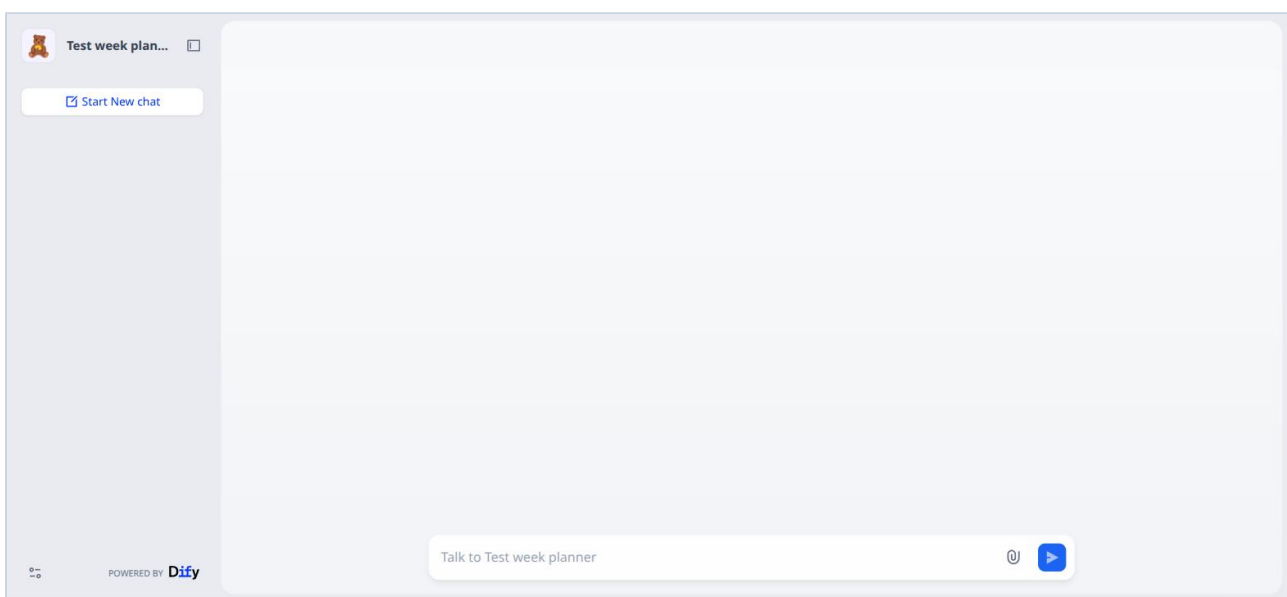
The agent is published

2. Click **Access Point** in the left-hand menu. You see the **Web App** with the **Access URL** and the **Backend Service API**.



The Access Point screen

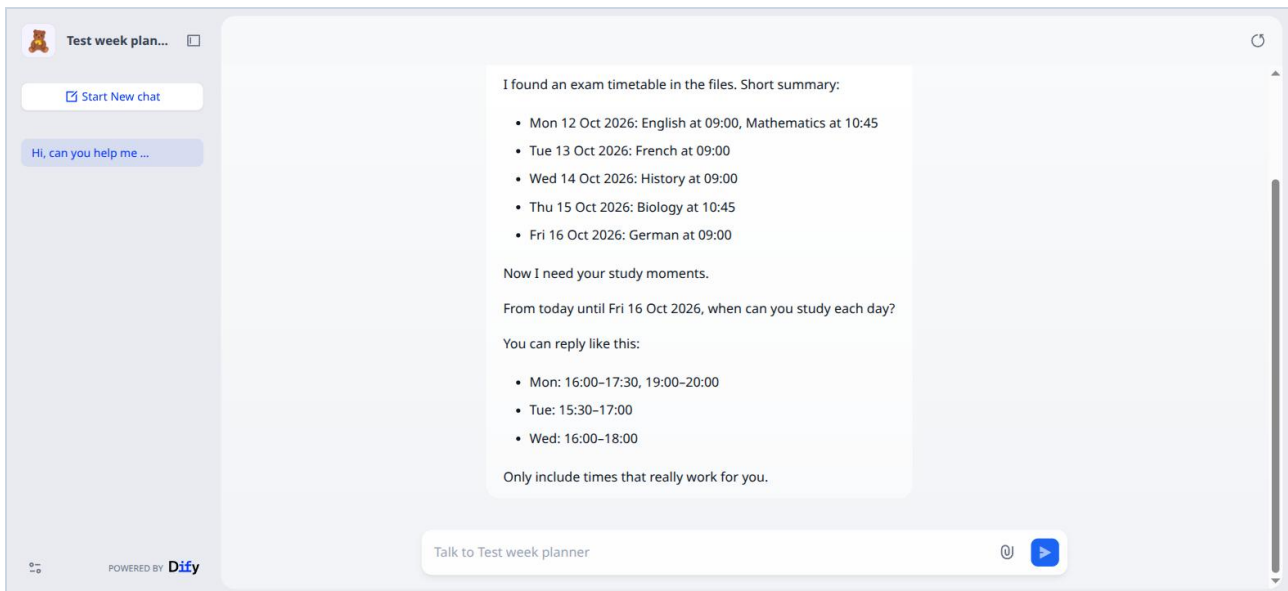
3. Click **Open** behind the URL. The webapp opens in a new tab: an empty chat window with the agent's name.



The webapp, as a student sees it

4. Type a message to check that the published version works.

Message	What you type
1	Hi, can you help me plan?

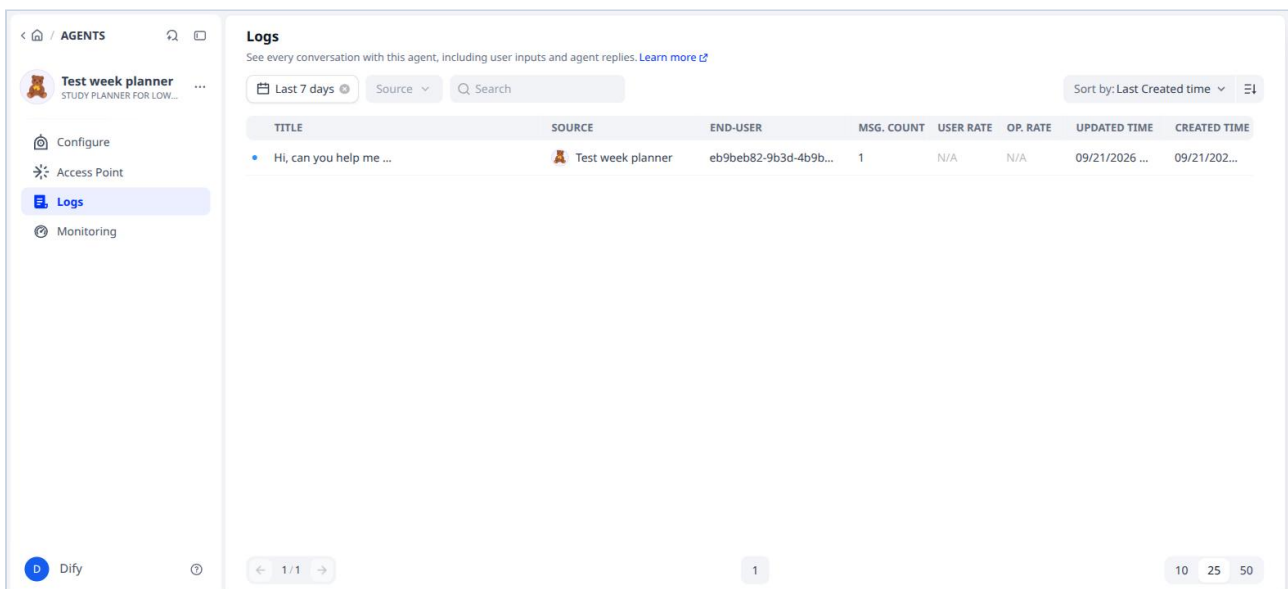


The conversation in the webapp

If you change the instruction later, the webapp only changes after you click **Publish update** again. Until then it says “Unpublished changes” at the bottom.

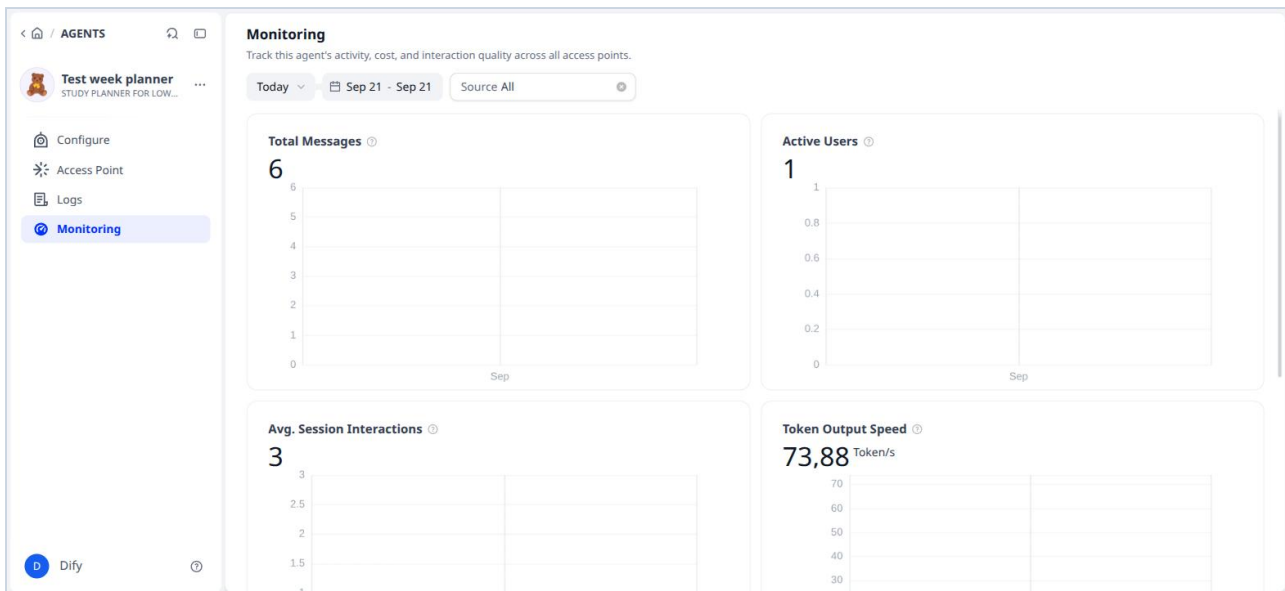
Step 8. Look at the logs

1. Click **Logs** in the left-hand menu. Every conversation through the webapp is listed here, with the first sentence as title, the number of messages and the time.



Logs with the conversation from the webapp

2. Click **Monitoring** for the counters: number of messages, active users, average length of a conversation and the speed of the model.



Monitoring of the agent

Conversations from the PREVIEW tab are not in Logs; only conversations through the webapp or the API. If you want to see a test conversation again, do the test in the webapp.

Checking that it worked

- Under Agents the Test week planner is listed without the DRAFT label
- The model is gpt-5.4 (or another compatible model), without the Incompatible label
- Under FILES is exam-timetable-example.csv, under TOOLS is CurrentTime
- In PREVIEW the agent asks its questions one at a time and only comes up with a plan after the third question
- Below the plan are two links: study-plan.xlsx and study-plan.ics
- The Access URL opens a working chat

When things go wrong

What you see	What is going on
Label Incompatible next to the model	This model cannot drive the Agent Console. Choose gpt-5.4 or higher, or a Gemini model (step 2)
The agent asks all questions at once	The sentence "Ask these questions one at a time" is missing from the instruction
The agent asks for the timetable while the file is there	The file name in the instruction does not match the uploaded file, or the file is not there yet (step 4)
The plan puts blocks at moments you did not give, or leaves study moments unused	The model interprets the rules; always check the plan against the rules (see also Make it yours). In our test the agent used 16:00 to 17:30 of the 16:00 to 18:00 you gave, so 30 minutes stayed unused each day
Only study-plan.xlsx appears, no .ics	The agent is still busy; wait until Stop responding is gone. If the file does not come, ask "Also make the ics file"
The agent does what the student asks against a rule	Normal: the rules are text the model interprets, not code. Make the rule explicit ("not negotiable") or accept it, see the green box at step 6
An empty answer bubble that stays	Beta behaviour: the agent stopped without text. Click the "retry" arrow under the message, or start a new conversation
The conversation in PREVIEW is not in Logs	Normal; only webapp and API conversations are logged (step 8)
The credits drop fast	Every conversation costs tens of thousands of tokens. Set OpenAI to Usage priority: API key, or test less often
"Upgrade to create more apps"	Sandbox limit of five apps; agents count. Delete an old app in Studio

Make it yours

The timetable is invented and the planning rules are an example. This chapter helps you rebuild the agent for your school, within the privacy frame at the start of this guide. From easy to hard.

Change	Where	Difficulty	What you get for it
Tone and form of address (formal, shorter, with emoji)	Step 3, first paragraph of the prompt	easy	An agent that sounds like your form tutors
Your planning rules	Step 3, heading Planning rules	easy	A plan that matches the study skills lessons you already give
The real exam timetable of a class	Step 4, replace the file	easy	A planner that is ready for the coming test week
Only an Excel, no calendar file (or the other way round)	Step 3, heading What you deliver	easy	Less work for the agent, so faster and cheaper
A third question: "which day do you want to keep free?"	Step 3, heading What you need	medium	A plan with room for sport or a job
A student who talks back gets their way	Step 3, extra rule (see the green box at step 6)	medium	Less discussion, more ownership for the student
A study planner per subject as an extra file	Step 4, more files	medium	Blocks with precise material ("section 3.2 and the glossary")
A tool that fetches the timetable from the school's timetable system	Step 5, MCP or API tool	hard	No more CSV to make; only possible on a school environment with the right connections
Running on your school's approved platform	Export DSL, import, reconnect the model	hard	The route that is compulsory for real students

Three assignments

1. **Your rules.** Ask two form tutors which three planning rules they give their students. Put those in the prompt instead of ours and run the same test conversation (step 6). Compare the two plans: which one would a student actually follow?
2. **Find the flaw.** Take the plan from your test and check it rule by rule: is every subject there at least twice, is the last time the day before the test, are there no blocks outside the given moments? Write down what is wrong and adjust the prompt until it is right. This is the core of working with agents: you remain the checker.
3. **One more tool.** Add the **Code Interpreter** tool (step 5) and ask the agent to make a bar chart of study time per subject. Then look under Commands executed at what it did. Remove the tool again if you do not need it.

Checklist before you use your own material

- [] No conversation with a real student goes through Dify cloud; only you or a colleague as the “student”
 - [] For real students: the agent runs on your school’s approved platform, the impact assessment is done and parents are informed
 - [] The timetable you upload contains no student names or class lists
 - [] The boundaries are in the prompt: no grades, judgements or predictions
 - [] Someone checks a handful of plans against the rules before students get the link
 - [] There is one owner of the agent who maintains the prompt and republishes
-

Housekeeping

What Dify keeps. Every conversation through the webapp is in **Logs**, including what the student typed. For the prototype with you as the student that is no problem. For real students this is the reason Dify cloud is not allowed: you cannot delete per student and the retention period cannot be set. On a school environment a retention period belongs with the agent.

Costs. The Agent Console is the most expensive way of building in Dify. A workflow from guide 1 makes one model call per run; this agent makes ten to twenty per conversation, plus writing and running programs. Count on 5 to 15 cents per complete conversation with gpt-5.4 through your own key. So put a limit on the key before a class of thirty students uses it.

Beta. The Agent Console can change from one week to the next: other buttons, other models that are or are not compatible, other behaviour with files. Keep the prompt outside Dify as well (it is in this guide), so you can recreate the agent in five minutes.

From prototype to your school environment. Via the three dots next to the agent’s name (**Export DSL**) you save the agent as a file and import it into another Dify account via **Create → Import DSL file**. The model has to be reconnected there and the file under FILES has to be uploaded again (an agent export lists it but does not contain it); the prompt and the tools come along. The export of this agent is in the folder of this guide as `test-week-planner.dify.yml`.

Licence: CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0/>). You may copy, adapt and reuse this document, including at your own school, provided you credit the source (Guido van Dijk, LeX Consultancy B.V., <https://git.lexconsultancy.nl/guido/dify-guides>) and share your adaptation under the same licence. The LeX Consultancy logo is excluded; product names are trademarks of their owners. Schools, people and documents in the examples are fictional.

Owner and contact: Guido van Dijk, LeX Consultancy B.V. · guido@l-e-x.nl

