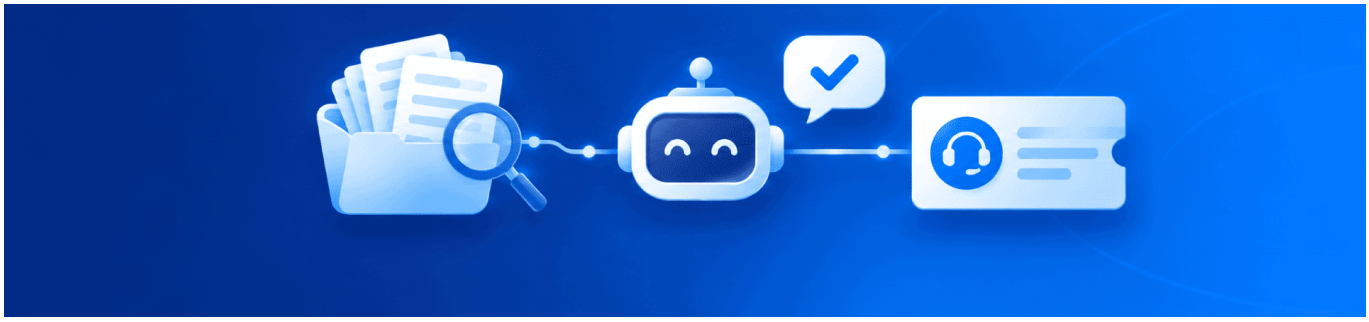
An AI support assistant that reads the docs: from half to 80% correct first replies

How the documentation of a product reaches an AI assistant through an MCP endpoint, how the support team grades every reply, and why the quality of the answers is the quality of the documentation.



Derafu · 03/11/2026 · #ai, #mcp, #documentation

An AI support assistant that reads the docs: from half to 80% correct first replies

Derafu DEV builds software. It does not sell products or deal with customers: other companies of the Derafu group do that, and they run the products that this software is built for. This post is about how some of those products answer their support tickets, and about the software behind it.

For more than three months, the first reply to the support tickets of the product that receives the most of them has been written by an AI assistant. The same assistant also answers the tickets of other products. It does not answer from memory: it reads the documentation. The support team reviews every ticket afterwards and grades the reply. The numbers in this post come from that busiest product, which has enough tickets to say something representative. Today about 80% of its replies are correct, which here means that they are enough to close the ticket. At the start, about half of them were.

The software part is ours: Derafu Content and its MCP plugin. The numbers and the way of grading belong to the support team.

One thing before the details: the quality of the assistant's answers is the quality of the documentation. Derafu Content does not improve an answer. It exposes what already exists, so a page that is missing, wrong or out of date becomes a reply that is missing, wrong or out of date. Almost everything in this post follows from that.

The wall the support team ran into

A support team answers the same kinds of questions again and again, and the answer is often written somewhere in the documentation. The documentation exists, but a customer in the middle of a problem does not go looking for it. They write a ticket.

The idea was to let an assistant do the looking and answer with what the documentation says. The hard part is not the assistant. It is making the documentation something that an assistant can read, and knowing, ticket by ticket, whether its answers are any good. That is where our software comes in.

The idea: the documentation is one folder

The documentation of that product is a folder of Markdown files: documentation, FAQs, courses and blog posts, around 500 files in total. It is published as a website with Derafu Content, the same software that runs this site. There is no database and no build step.

The same files are also an interface for machines. Every page can be asked for as Markdown, PDF or JSON, and Content has a plugin that publishes the whole site as an MCP server. MCP (Model Context Protocol) is the protocol that AI clients use to call tools: a server just says which tools it has. That server has three:

Tool	What it does
`get_content`	Returns the full Markdown and the metadata of one page.
`search_content`	Searches the docs, the blog, the FAQ, the pages and the academy, through the search engine that is connected.
`ask`	Returns an answer written by an LLM from the content. It is off until you enable it and give it an LLM.

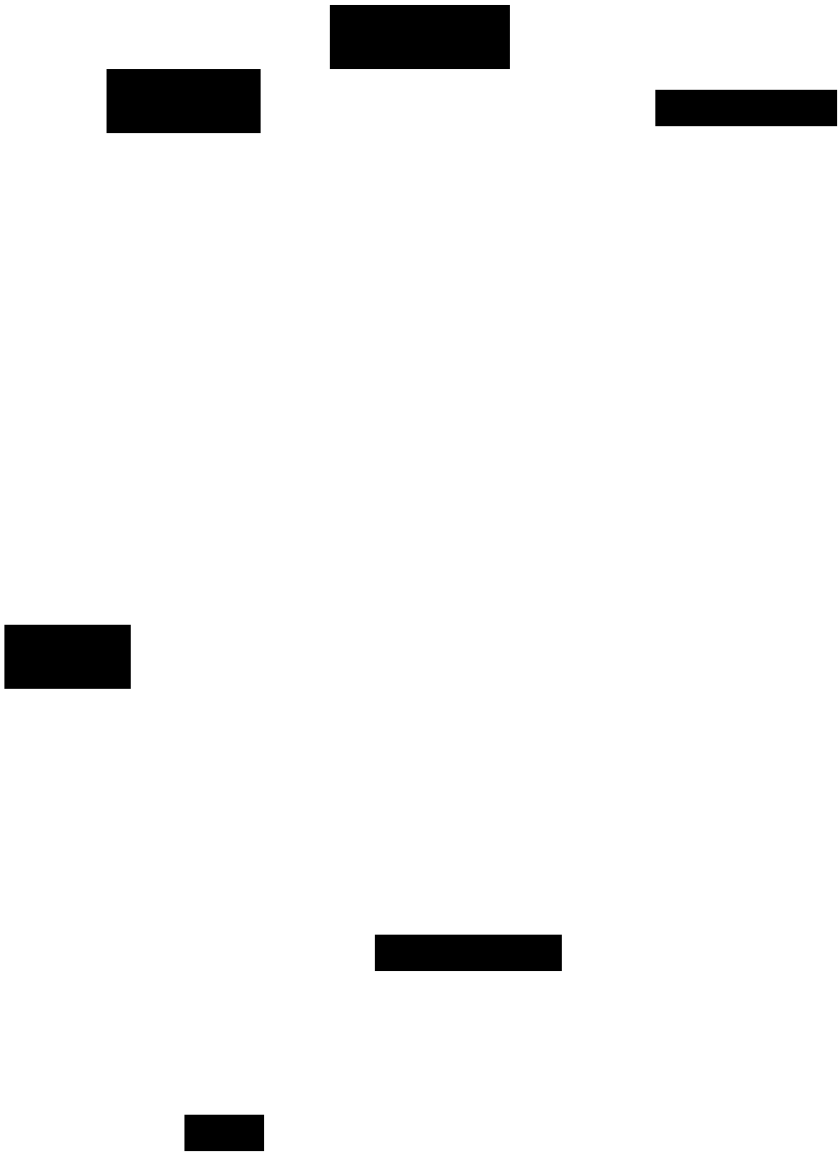
How it is set up

The endpoint is one route, `POST /api/content/mcp`, and it needs one package that you add yourself, the official `mcp/sdk`. The rest is configuration. This is its shape, with the services left generic:

```
parameters:
    derafu.content.config:
        plugins:
            docs: ~
            faq: ~
            blog: ~
            academy: ~
            search:
                # %%s is a %s placeholder, escaped for the container.
                url:
```

```
'https://search.example.com/api/search?collection=%%s&base_url=%%s&text=%%s'
                collection: 'my-product'
                base_url: 'https://www.example.com'
                llm_url: 'https://llm.example.com'
                llm_model: 'my-model'
                llm_api_key: '%env(LLM_API_KEY)%'
            mcp:
                ask:
                    enabled: true
```

Content is one piece of a larger chain. This is how all of it connects:



- **Derafu Content** publishes the folder and exposes it. It indexes nothing and it writes no reply.
- **The search engine** is Qdrant. The API export lists every page as one JSON list, with a checksum for each page, and it is meant for indexing pipelines. A pipeline splits each page into chunks, turns each chunk into a vector and stores it in Qdrant, and it can skip the pages whose checksum did not change. How the pages are split and stored has its own details, and it affects what the assistant finds.
- **The LLM** is reached through OpenRouter, which has a good cost and benefit for the team at this moment. Content works with any API that is compatible with the OpenAI chat format, and the team has also tried local models served with Open WebUI, with decent results.
- **The tickets** are processed by an n8n workflow. It is the piece that orchestrates a ticket with the content and the LLM.

How the support team measures it

Every ticket is reviewed by a person of the support team, who grades the reply of the assistant on three levels:

- **Correct.** It answers what the user asked, and the answer or the steps in it should be enough for the user to solve the problem.
- **Medium.** It is incomplete: a person has to step in.
- **Incorrect.** It is plainly bad: a person has to answer from zero.

Level	At the start	Today
Correct	about 50%	about 80%
Medium	about 30%	about 15%
Incorrect	about 20%	about 5%

The loop

When a reply is medium or incorrect, the team improves the documentation and then keeps going. That is the whole method: the next ticket about the same topic meets a better page.











They credit this loop for the change in the numbers. Each reply that is not correct points at a page to improve, and the pages are plain Markdown files. They have repeated it for more than three months.

What it costs, and what the number does not say

- **The quality of the answers is the quality of the documentation.** The model, the search engine and the workflow matter, but a good one cannot answer from a page that does not exist. If your documentation is thin, ambiguous or out of date, expect the same from the assistant. Your number will depend on your documentation more than on anything in this post.
- **A person still reads every ticket.** The 80% is not 80% less work. About 20% of the replies still need a person, and the 5% that are incorrect need an answer from zero.
- **The grade is internal and it is not a benchmark.** It is made by the company that runs the products, on support questions. Most of the volume comes from one product, which receives enough tickets to be representative, although the assistant also answers the tickets of others.
- **Search depends on how the content is indexed.** A page can be good and still be hard to find if it is split or stored badly. Three things are worth checking: the size of the chunks and how much they overlap, whether the headings travel with the text of each chunk, and whether the embedding model works well in the language of your documentation.
- **There are more services to run.** A search engine, an LLM and a workflow tool have their own cost and their own ways to fail. They are choices that you make; Content only connects to them.
- **The endpoint reads, it does not write.** An agent can search and read your content, but it cannot create or edit it. The plugin has no authentication and no rate limiting of its own, so you protect the endpoint in your HTTP stack.
- **Edits are not instant.** The tree of pages is cached for 60 seconds by default, so an edit shows up on the site and in `get_content` within about a minute. In the search results it shows after the next indexing run, which is a separate step.

If you want to try it

1. Write the documentation as Markdown files in folders, or move it there. Spend your first effort here.
2. Enable the plugins that you use, and the `mcp` plugin.
3. Add `mcp/sdk`, a search engine and, if you want answers, an LLM through any API that is compatible with the OpenAI chat format. Something has to orchestrate them: the team uses n8n.
4. Before you automate anything, grade by hand: decide your levels and review every reply.
5. When a reply is medium or incorrect, improve the page.

In short

The part that took the replies from half to 80% was not a feature. It was a habit: every reply that was not correct became an edit to the documentation. Derafu Content makes that cheap, because the

documentation is a folder of files, and it makes the same files readable by an assistant. What it cannot do is write the pages for you.

The Derafu Content page has an overview. The reference of the MCP plugin and the search plugin has every option.

Last updated on 05/11/2026