

# USING AI AND LLMS IN DOCS-AS-CODE PIPELINES

*Continuous Documentation Regulars,  
24th January 2024 ([Recording](#))*

Jonathan Algar  
Senior Technical Writer, OutSystems

# WHY DOCS-AS-CODE?

*“Continuous Integration and Delivery is commonplace for software. Let’s benefit from the same principles and apply them to documentation as well: Keep it up to date, run automated tests, collaborate as teams, roll it to production seamlessly.” —Continuous Documentation Regulars*

# WHY DOCS-AS-CODE?

*“Continuous Integration and Delivery is commonplace for software. Let’s benefit from the same principles and apply them to documentation as well: Keep it up to date, run automated tests, collaborate as teams, roll it to production seamlessly.” —Continuous Documentation Regulars*

OutSystems Technical Knowledge tooling:  
VSC, Markdown, Vale, Git, GitHub actions

# WHY LLMS?

Team experiments with LLMs for content production and editing had good results in two broad categories:

- 1.
- 2.

# WHY LLMS?

Team experiments with LLMs for content production and editing had good results in two broad categories:

1. Automating routine writing tasks.
- 2.

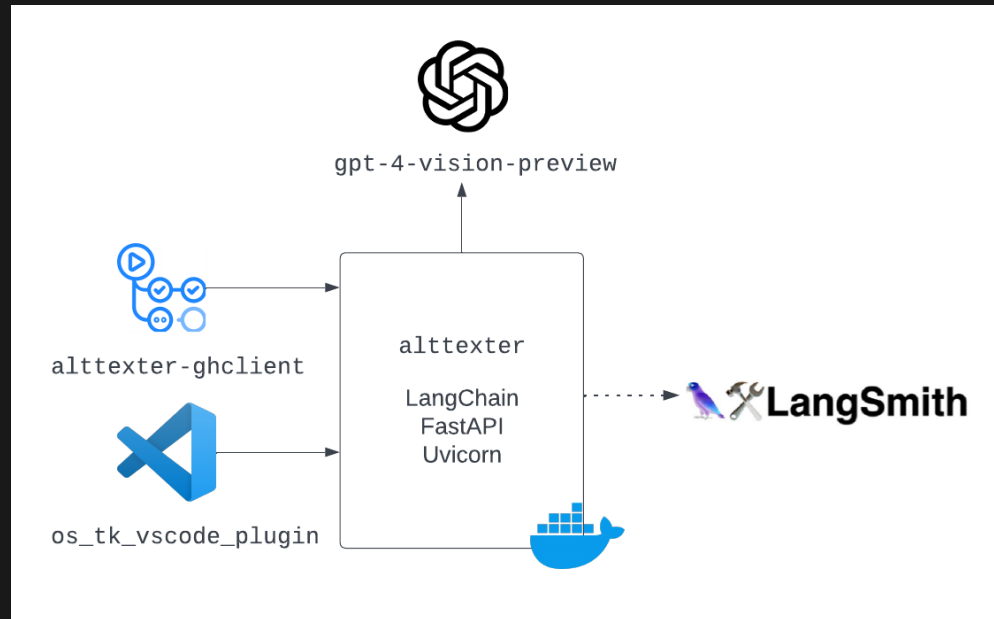
# WHY LLMS?

Team experiments with LLMs for content production and editing had good results in two broad categories:

1. Automating routine writing tasks.
2. Providing suggestions for bringing content into compliance with the OutSystems Style Guide.

# APPLYING LLMs TO DOCS-AS-CODE

## USE CASE #1: ADD MISSING IMAGE ALT TEXT



- Required per [OutSystems Style Guide](#) but often missing in contributions.
- EU Accessibility Act 2025.

# alttexter REQUEST SCHEMA


AUTHORIZATIONS: >	<i>APIKeyHeader</i>
REQUEST BODY SCHEMA:	application/json
text	string (Text) Article containing markdown formatted text.
required	
images	object (Images) Default: {} Local images defined in markdown article encoded in base64 format.
image_urls	Array of strings (Image Urls) Default: [] Image URLs defined in markdown article.

- Defined as Pydantic object.
- Information in request is processed and structured in the prompt for the LLM.



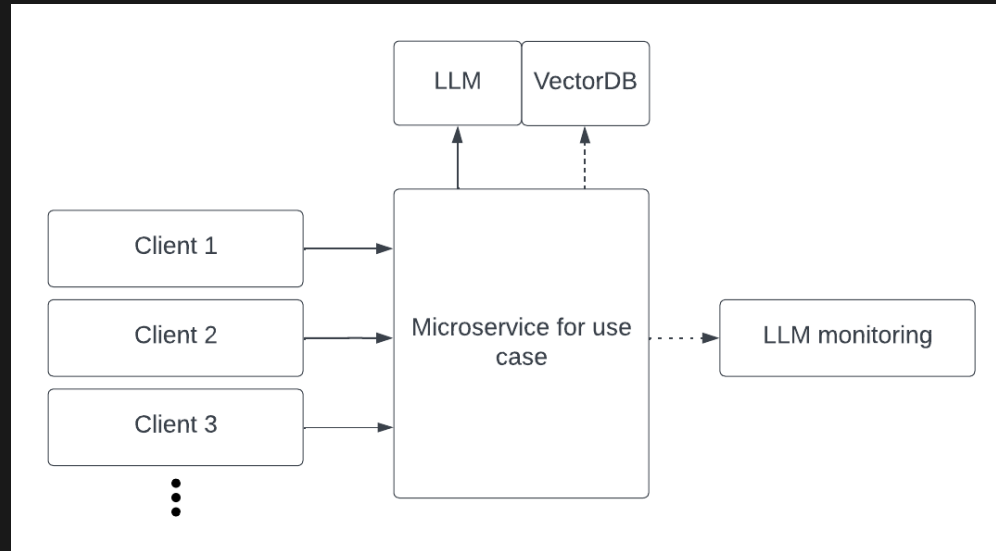
# alttexter RESPONSE SCHEMA

RESPONSE SCHEMA: application/json

images  required	Array of objects (Images) A list of images.
Array [	
name required	string (Name) File name of the image including path or URL.
title required	string (Title) Title of the image.
alt_text required	string (Alt Text) Concise alternative text for the image.
]	
run_url	string (Run Url) LangSmith trace URL.

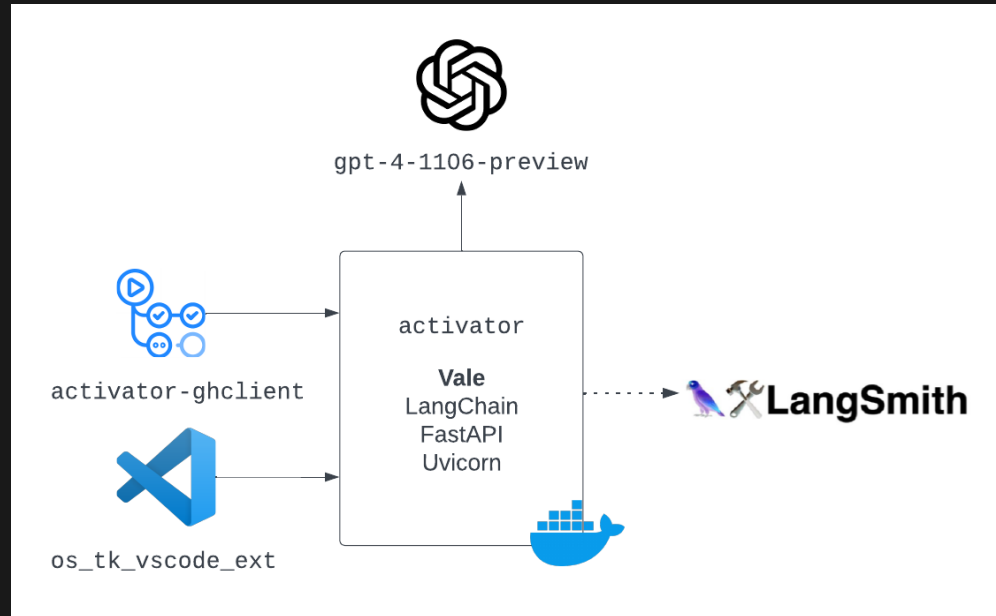
What can a client do with this? See [jonathanalgar/docs-demo/pull/1](https://github.com/jonathanalgar/docs-demo/pull/1)

# GENERAL APPROACH TO TAKING EXPERIMENTS TO PRODUCTION



- Build microservices for prioritized use cases.
- Microservices consumed by different clients: GitHub actions, VSCode, Rider,...

# USE CASE #2: FIX PASSIVE VOICE



- Writing in the active voice is hard. Particularly if you don't have experience in technical writing.
- Also required per [OutSystems Style Guide](#).

# activator REQUEST SCHEMA

<u>AUTHORIZATIONS:</u> >	<i>APIKeyHeader</i>
REQUEST BODY SCHEMA:	application/json
→ text required	string (Text) Text.

Very simple!

# INTEGRATING VALE WITH A LLM

```
1 @app.post (
2     "/activator",
3     response_model=ExtendedActivatorResponse,
4     responses={
5         422: {"model": ErrorResponse, "description": "Valid
6         500: {"model": ErrorResponse, "description": "Inter
7     }
8 )
9 def activator_text (
10     request: ActivatorRequest = Body(...),
11     token: str = Depends(get_api_key)
12 ):
13     """Endpoint to process activator requests."""
14     try:
15         text = request_text
```

(via [jonathanalgar/activator/main.py](#))

# INTEGRATING VALE WITH A LLM

```
14         elif .
15             text = request.text
16
17             # Preprocessing
18             if (is_valid_notebook(text)):
19                 text = keep_only_markdown_cells(text)
20
21             # Extract passive sentences using Vale
22             passive_sentences = process_with_vale(text)
23
24             if not passive_sentences:
25                 return ExtendedActivatorResponse(violations=[],
26
27             # Send to LLM
28             active_response, run_url = activator(text, passive_
29             if active_response.violations:
```

(via [jonathanalgar/activator/main.py](https://github.com/jonathanalgar/activator/main.py))

# INTEGRATING VALE WITH A LLM

```
20
21     # Extract passive sentences using Vale
22     passive_sentences = process_with_vale(text)
23
24     if not passive_sentences:
25         return ExtendedActivatorResponse(violations=[],
26
27     # Send to LLM
28     active_response, run_url = activator(text, passive_
29     if active_response is None:
30         raise Exception("Failed to generate active sent
31
32     return ExtendedActivatorResponse(violations=active_
33
34     except Exception as e:
35         handle_endpoint_error(e)
```

*(via [jonathanalgar/activator/main.py](https://github.com/jonathanalgar/activator/main.py))*

# INTEGRATING VALE WITH A LLM

```
22     passive_sentences = process_with_vale(text)
23
24     if not passive_sentences:
25         return ExtendedActivatorResponse(violations=[],
26
27     # Send to LLM
28     active_response, run_url = activator(text, passive_
29     if active_response is None:
30         raise Exception("Failed to generate active sent
31
32     return ExtendedActivatorResponse(violations=active_
33
34     except Exception as e:
35         handle_endpoint_error(e)
36 }
```

*(via [jonathanalgar/activator/main.py](#))*



# activator RESPONSE SCHEMA

RESPONSE SCHEMA: application/json

violations	Array of objects (Violations)
required	A list of all instances of sentences with passive voice in the article.
Array [	
original_sentence	string (Original Sentence)
required	The original sentence from the article containing instance(s) of passive voice.
revised_sentence	string (Revised Sentence)
required	The revised sentence in active voice.
clear_explanation	string (Clear Explanation)
required	A clear explanation of the choice of subject.
]	
run_url	string (Run Url)
	LangSmith trace URL.

What can a client do with this? See [jonathanalgar/docs-demo/pull/2](https://github.com/jonathanalgar/docs-demo/pull/2)

# NOW TO THE GROUP DISCUSSION...

---

## Contact:

- [github.com/jonathanalgar](https://github.com/jonathanalgar)
- [linkedin.com/in/jonathanalgar](https://linkedin.com/in/jonathanalgar)

Slides: [jonathanalgar.github.io/slides](https://jonathanalgar.github.io/slides)