

# Open Source DocOps

Lorna Mitchell, Redocly

# What is DocOps?

Documentation operations, allowing docs to be created, maintained and published collaboratively and in an efficient manner.

# Docs-as-code

DocOps builds on Docs-as-code:

- source control
- text-based markup
- continuous integration
- local tools

# Open Source

Open source means freedom.

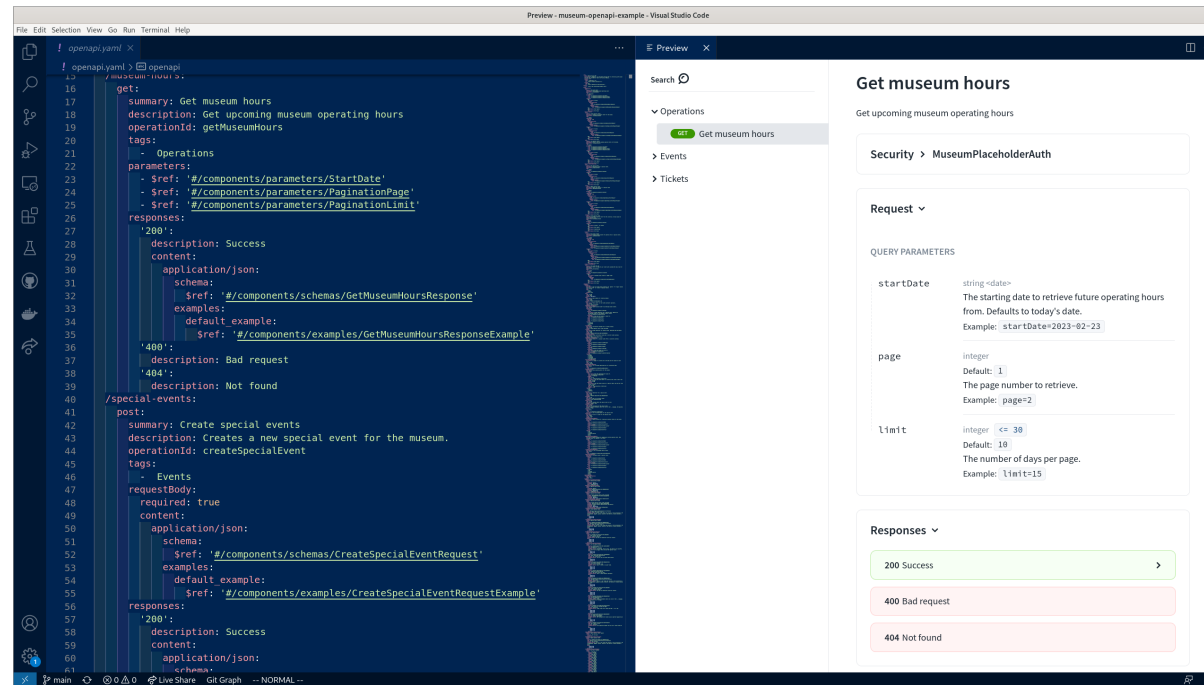
Open source also means:

- not having to rebuild the same tools over again
- run it where you want, CI, local
- we collaborate to improve our shared tools
- no vendor lock-in

# Tool the Docs

# Preview changes

All documentation contributors should be able to see what they are doing.



The screenshot displays the Visual Studio Code interface with an OpenAPI specification on the left and its preview on the right. The OpenAPI spec defines a 'get' operation for 'Get museum hours' with parameters for 'startDate', 'page', and 'limit'. The preview on the right shows the 'Get museum hours' operation, including security requirements, request parameters, and response codes (200 Success, 400 Bad request, 404 Not found).

```
16 get:
17   summary: Get museum hours
18   description: Get upcoming museum operating hours
19   operationId: getMuseumHours
20   tags:
21   - Operations
22   parameters:
23     - $ref: '#/components/parameters/StartDate'
24     - $ref: '#/components/parameters/PaginationPage'
25     - $ref: '#/components/parameters/PaginationLimit'
26   responses:
27     '200':
28       description: Success
29       content:
30         application/json:
31           schema:
32             $ref: '#/components/schemas/GetMuseumHoursResponse'
33           examples:
34             default_example:
35               $ref: '#/components/examples/GetMuseumHoursResponseExample'
36     '400':
37       description: Bad request
38     '404':
39       description: Not found
40   /special-events:
41     post:
42       summary: Create special events
43       description: Creates a new special event for the museum.
44       operationId: createSpecialEvent
45       tags:
46       - Events
47       requestBody:
48         required: true
49         content:
50           application/json:
51             schema:
52               $ref: '#/components/schemas/CreateSpecialEventRequest'
53             examples:
54               default_example:
55                 $ref: '#/components/examples/CreateSpecialEventRequestExample'
56       responses:
57         '200':
58           description: Success
59           content:
60             application/json:
61               schema:
```

# Link checking

Automatically "click" every link and check it is valid.

Check links in source or output format:

- Source markup format (I use `m1c`)
- An HTML link checker, run after build

# Link checking strategies

The internet goes wrong sometimes.

- restrict scope of checks (internal links, changed files)
- weekly check all links for decay or other problems

TIP: Someone else's downtime should never block your build/release.



# Validation

Check for syntax errors.

Catches errors that aren't visually obvious.

The machines can help us!

# Validation tools

Docs build tools may do this as standard. Or try:

- Markdown:

<https://github.com/DavidAnson/markdownlint>

- ReStructuredText: Sphinx includes this feature

- OpenAPI: <https://github.com/Redocly/redocly-cli/>

# Formatting

Valid is one thing, consistent and readable are another.

Formatters adjust:

- newlines and spacing
- indentation
- line length
- bullet/table syntax

# Prose linting

Quality assurance for the written word.

Catch common mistakes with <https://vale.sh>

- repeated words
- correct case and spelling for product names
- dictionary of your words
- ... and anything else you can think of

# Workflows and environments

Make the tools work with your process.

# Source control branching

GitHub flow:

- small changes
- each in their own branch
- manageable to review

# Continuous integration

Apply the tools to every pull request:

- never miss running any check/test
- build previews to assist reviewers

Maintain documentation quality.

# Feedback loops

Opening a pull request is a big feedback loop.

We need small feedback loops:

- all tools available locally
- IDE integration a bonus (but avoid lock-in)
- covered by setup docs



# Open Source DocOps

Move fast and don't break anything!

# Resources

- <https://lornajane.net>
- <https://redocly.com>
- <https://github.com/becheran/mlc>
- <https://github.com/DavidAnson/markdownlint>
- <https://github.com/Redocly/redocly-cli/>
- <https://vale.sh/>