Lifting the Documentation Burden

Texas Linux Festival 2025

Michael Troutman, Documentation Specialist, LINBIT

Introduction

▓▓ Who I am

Documentation Specialist at LINBIT since January 2022

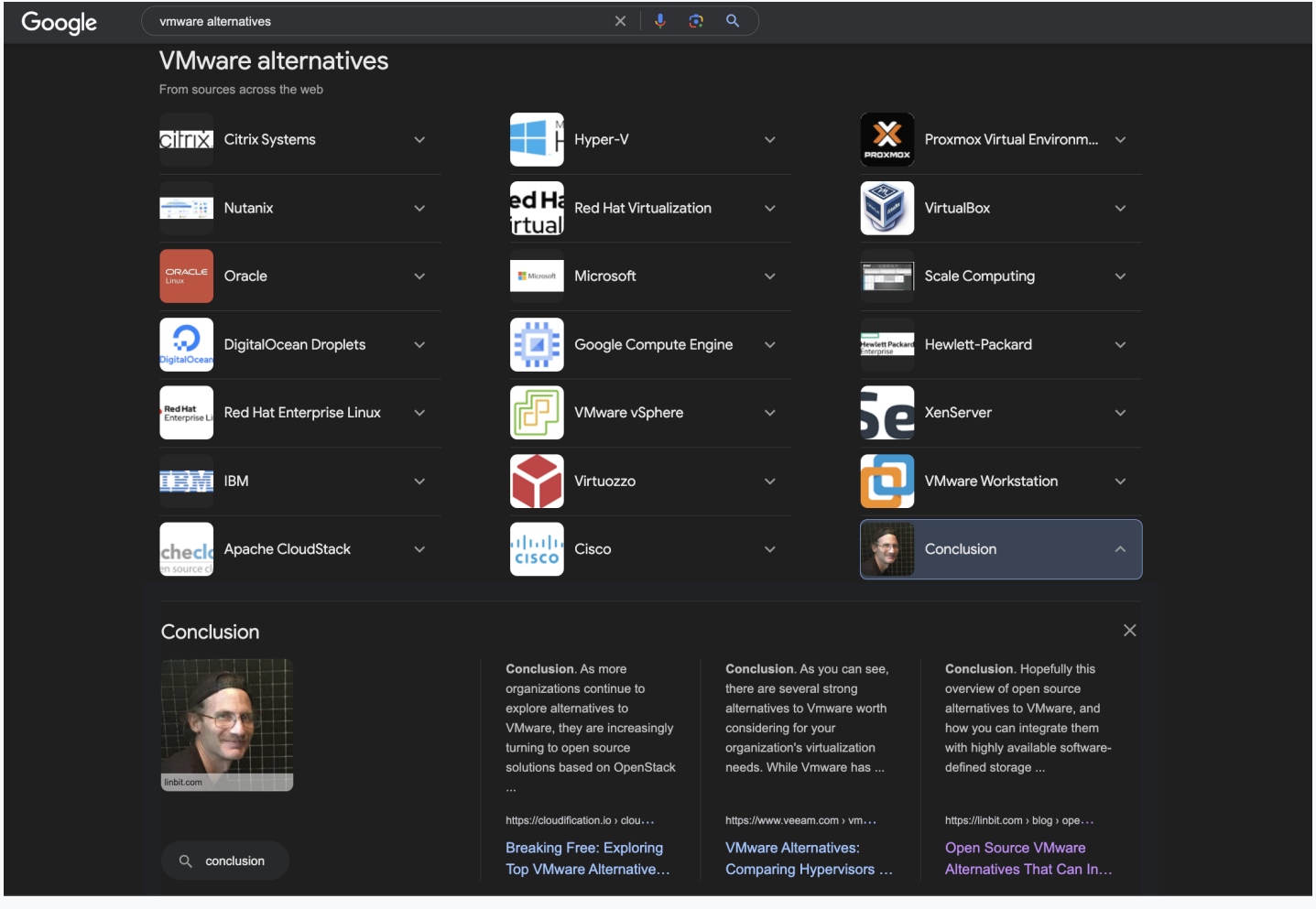
▓▓ What this talk is about

How you can improve documentation by using open source Linux tools

▓▓ Who is this talk for

Hopefully anyone who cares about quality and consistency in documentation but especially relevant to small teams, even those without a dedicated documentation specialist or technical writer

Why should you care about what I say?



The open source tools I will focus on

▓▓ Vale prose linter

- https://vale.sh/
- https://github.com/errata-ai/vale

▓▓ *Red Hat supplementary style guide for product documentation*

- https://redhat-documentation.github.io/supplementary-style-guide/
- https://github.com/redhat-documentation/vale-at-red-hat/

▓▓ GitLab

- https://about.gitlab.com/
- https://gitlab.com/rluna-gitlab/gitlab-ce/ (Community Edition)

Vale

Red Hat Suplementary Style Guide

GitLab

Don't forget the soft skills

The motivation for using these tools

▓▓ For your users

- Quality and consistency improve the user experience
  - Align documentation with what users might already be familiar with

▓▓ For your team

- Appeal to an outside authority for settling questions or disagreements within teams about writing
- Appeal to Linux tinkerers (the nerd factor)

> **! IMPORTANT**
> Check in with stakeholders before adopting a style guide.

▓▓ For the community

- Using (and contributing to) open source projects builds community

Spend more time doing more meaningful work

User interface elements

Use bold text for all graphical user interface (GUI) element names, including menus, menu items, buttons, dialog boxes, and windows. Use bold text for the element name if the name appears in the GUI, even if the element is not clickable.

Example AsciiDoc

```
On the *Installed Operators* page, click *Metering*.
```

If an element is not labeled in the GUI, refer to the element by a generic description and do not use bold text. For example, if a search field is not labeled in the GUI, write it as "the search field", not "the **Search** field".

— Red Hat supplementary style guide for product documentation

```
$ lbv "On the *Installed Operators* page, click on *Metering*."

 stdin.txt
 1:36  warning  Consider using 'click'          RedHat.TermsWarnings
                rather than 'click on' unless
                updating existing content that
                uses the term.
```

Introducing the Vale prose linter

> A markup-aware linter for prose built with speed and extensibility in mind
>
> — Arch Linux package manager description

- Product page: https://vale.sh/
- Documentation: https://vale.sh/docs/
- GitHub codebase: https://github.com/errata-ai/vale/

▓▓ Benefits of using a prose linter

- Enforce consistency across documentation.
- Align the writing of different writers.
  - Make allowances for content types and settings, for example, a blog article compared with a user guide.
- Appeal to an outside authority to settle questions or disagreements.

▓▓ Vale limitations

Vale is just a linter.

However, using Vale can help you become more mindful about your writing. Over time, this can improve your writing.

For example, may vs. might:

```
lbv "After installing the program, you may notice a new desktop icon."

 stdin.txt
 1:35  warning  Consider using 'might' or       RedHat.TermsWarnings
                'can' rather than 'may' unless
                updating existing content that
                uses the term.

✖ 0 errors, 1 warning and 0 suggestions in stdin.
```

▓▓ Installing Vale

Vale is available as a package in many Linux distributions.

...or use it within a container for portability.

```
FROM jdkato/vale:v3.12.0

RUN mkdir /vale_app

WORKDIR /vale_app

COPY vale.ini /vale_app/.vale.ini

# a tool to convert Vale output to a GitLab CI pipeline friendly format
COPY tools/vale2climate /bin/vale2climate

# pull latest packaged style rules configured in `.vale.ini`
RUN /bin/vale sync

# copy LINBIT styles and vocabularies to expected container image dir
COPY styles/ /vale_app/styles/

ENTRYPOINT ["/bin/vale", "--config=/vale_app/.vale.ini"]
```

— A Dockerfile within a GitLab version-controlled project

▒▒ Building the container image:

```
docker build --tag lb-vale .
```

▒▒ Verifying the container image:

```
docker images --filter reference=lb-vale
```

── [finished] ──

```
REPOSITORY   TAG       IMAGE ID       CREATED      SIZE
lb-vale      latest    244b3c16ed20   7 days ago   218MB
```

Using the Vale prose linter

Run a Vale linting command from a container based on the container image:

```
docker run --rm -it -v $PWD:$PWD -w $PWD lb-vale <file>
```

Example output

```
 testinvalid.adoc
 2:1    error  Use 'many' or 'much' rather     RedHat.TermsErrors
               than 'a lot'.
[...]
 384:1  error  Use 'want' rather than 'would   RedHat.TermsErrors
               like'.
 388:1  error  Use 'zero' rather than 'zero    RedHat.TermsErrors
               out'.

✖ 207 errors, 0 warnings and 0 suggestions in 1 file.
```

▒▒ Vale notification levels

- Error
- Warning
- Suggestion

> **TIP**
>
> Use errors as blockers in a CI pipeline to prevent merging a commit in your version control platform.

▒▒ Using Bash aliases for running Vale

```
# report errors only
alias lbve='docker run --rm -it -v $PWD:$PWD -w $PWD lb-vale --minAlertLevel=error'

# report errors and warnings (default)
alias lbv='docker run --rm -it -v $PWD:$PWD -w $PWD lb-vale'

# report errors, warnings, and suggestions
alias lbvs='docker run --rm -it -v $PWD:$PWD -w $PWD lb-vale --minAlertLevel=suggestion'
```

Configuring Vale

You configure Vale by editing a configuration file, `.vale.ini`.

You can do things such as:

- Configure which style rule packages to use.
  - Synchronize with latest available by using a runtime command.
- Apply certain rules only for certain kinds of files.
  - For example, Markdown or AsciiDoc.
- Configure Vale to ignore certain kinds of content.

```
# IgnoredScopes specifies inline-level HTML tags to ignore.
# These tags may occur in an active scope (unlike SkippedScopes, skipped entirely) but their content still will not raise any alerts.
# Default: ignore `code` and `tt`.
IgnoredScopes = code, tt, img, url, a, body.id

# SkippedScopes specifies block-level HTML tags to ignore. Ignore any content in these scopes.
# Default: ignore `script`, `style`, `pre`, and `figure`.
# For AsciiDoc: by default, listingblock, and literalblock.
SkippedScopes = script, style, pre, figure, code, tt, blockquote, listingblock, literalblock
```

> **NOTE**
>
> Vale converts files to HTML before linting them.

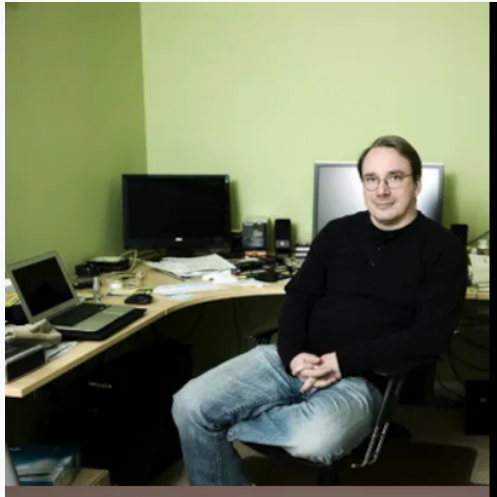
Creating Vale style rules

Vale style rules are the heart of using the Vale tool. Style rules are how you enforce consistency.

For example, you might create a style rule to enforce the correct use of a product name or term.

```
# Rule to flag as an error the first occurrence of DRBD in running (body) text
# without the registered trademark symbol immediately following it, so long as
# there was not an earlier occurrence of DRBD followed by the ® symbol
extends: conditional
message: |
  First occurrence of '%s' must be followed by '®' (text),
  '®' (Markdown), or '(R)' (AsciiDoc)."
# refer to: https://vale.sh/docs/topics/scoping/
scope:
  - paragraph & ~comment.line & ~comment.block & ~heading
level: error
ignorecase: false
# Set `limit` to `1` so that the rule is only triggered once.
limit: 1
first: \bDRBD\b
second: '(DRBD)(?:®|\®\;|\(R\))'
```

Vale style rule packages



```
$ qqwing --generate
. . . | . . . | . . .
9 6 2 | . . . | . 1 .
. 4 1 | 8 . . | . 2 6
-------|-------|-------
1 . 4 | . . . | . . .
. 8 . | . . 3 | 6 5 .
3 . . | . . . | 2 . .
-------|-------|-------
4 2 . | . . 5 | . 6 .
. . . | 1 . 8 | . . 3
. . 8 | 3 . . | 5 . .
```

```
$ grex -c 'DRBD®' 'DRBD&reg;' 'DRBD(R)'
^DRBD(?:&reg;|\(R\)|®)$
```

Creating Vale style rules might be intimidating to a newcomer.

▒▒ Using pre-made style rules

Use pre-made style rules to get started by installing a Vale style package, for example:

- Red Hat
- Microsoft
- Google

Other packages:

- `write-good` (nit pick your writing)
- `readability` (alerts related to, yes, readability)

▒▒ Getting started

https://github.com/errata-ai/packages

Excerpt from a Red Hat style rule

```
---
extends: substitution
ignorecase: true
level: error
link: https://redhat-documentation.github.io/vale-at-red-hat/docs/main/reference-guide/termserrors/

message: "Use '%s' rather than '%s'."
action:
  name: replace
# swap maps tokens in form of bad: good
swap:
  "(?<!-)healthcheck|health-check": health check
  "(?<!-)pm": PM
  "(?<!-|.|{)env": environment
  "(?<!-|I )am": AM
  "(?<!.)io": I/O
  [...]
  "a lot(?: of)?": many|much
  "best of breed|best-of-breed": best in class
  "bottle neck|bottle-neck": bottleneck
  [...]
  untar: extract
  unzip: extract|decompress
  [...]
```

▒▒ Verifying style rules

```
lbve fixtures/LINBIT/trademark_errors.md
```

> **NOTE**
>
> Fixtures are files that you can use to test Vale style rules.

```
fixtures/LINBIT/trademark_errors.md
9:35   error  First occurrence of 'DRBD'      LINBIT.Trademark_DRBD
              must be followed by '®'
              (text), '®' (Markdown), or
              '(R)' (AsciiDoc)."
15:1   error  First occurrence of 'LINBIT'    LINBIT.Trademark_LINBIT
              must be followed by '®'
              (text), '®' (Markdown), or
              '(R)' (AsciiDoc)."
19:19  error  First occurrence of 'LINSTOR'   LINBIT.Trademark_LINSTOR
              must be followed by '®'
              (text), '®' (Markdown), or
              '(R)' (AsciiDoc)."
```

Using Vale with version control software

This is where you can tick some "docs as code" check boxes:

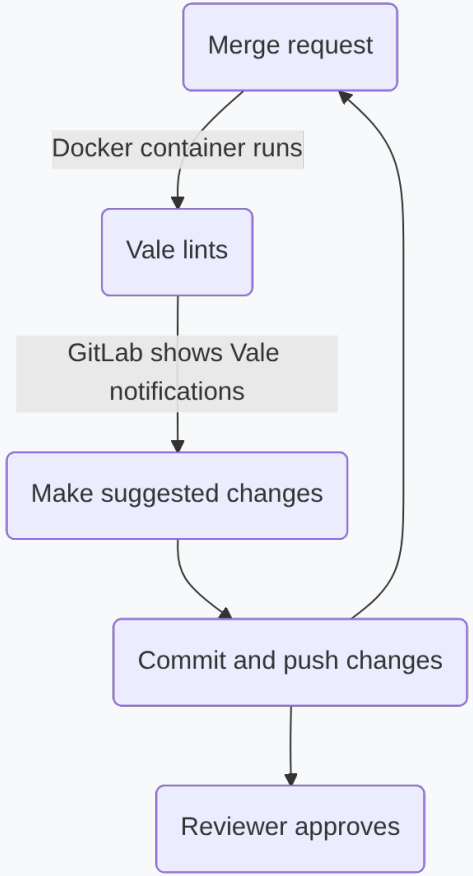
> [A] philosophy that you should be writing documentation with the same tools as code.
>
> ✓ Issue Trackers
> ✓ Version Control (Git)
> ✓ Plain Text Markup (Markdown, reStructuredText, Asciidoc)
> ✓ Code Reviews
> ✓ Automated Tests
>
> — https://www.writethedocs.org/guide/docs-as-code/

▓▓ Benefits of using Vale with version control software

- Automated prose linting saves time
- No need to worry about setting up Vale in different local environments
- Good fit for the review-suggest-refine process
- Introduce something new (Vale) in familiar context

Adding a lint step to a CI pipeline

You can add Vale to a CI pipeline that can make Vale prose linting part of a team's merge request review process.



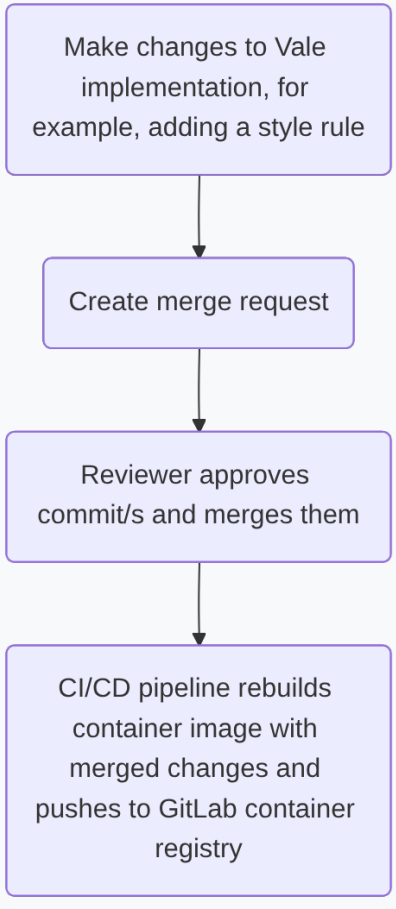
Setting things up in GitLab

In the code repositories where your documentation lives, you can add a `lint` job to a stage within CI pipelines:

```
lint:
  stage: build
  image:
    name: $LINBIT_DOCKER_REGISTRY/lb-vale:latest
    entrypoint: [""]
  rules:
    - if: $CI_MERGE_REQUEST_ID && $CI_MERGE_REQUEST_TARGET_BRANCH_NAME == 'master'
  script:
    - apk add --no-cache git
    - git fetch
    # use `tr` to sanitize variable in case there are multiple `.adoc` files to lint
    - ADOCFILES="$(git diff-tree --no-commit-id --name-only -r remotes/origin/${CI_MERGE_REQUEST_TARGET_BRANCH_NAME} ${CI_COMMIT_SHA} '*.adoc' | tr
'\n' ' ')"
    - |
      if [ -z "$ADOCFILES" ]; then
        echo "No .adoc files in this MR"
        echo -e '[\n  {\n    "description": "Merge request has no .adoc files",\n    "check_name": "ADOCFILES-variable-is-empty",\n    "fingerprint":
"0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0a",\n    "severity": "info",\n    "location": {\n      "path": "na",\n      "lines": {\n        "begin": 0\n      }\n
}\n  }\n]' > vale.json
        cp vale.json vale-climate.json
      else
        /bin/vale $ADOCFILES --config=/vale_app/.vale.ini --no-exit --output=JSON | tee vale.json
        /bin/vale2climate vale.json > vale-climate.json
      fi
  artifacts:
    paths:
      - vale.json
    when: always
    reports:
      codequality:
        - vale-climate.json
```

> **TIP**
>
> Use the `--no-exit` Vale command option to prevent Vale reported errors from blocking a merge.

Using a CI/CD pipeline to keep the Vale container image up-to-date



In a version control software repository, maintain your:

- Vale container image build file
- Vale configuration file
- Style rules, fixtures, and vocabularies

Then you can use a CI pipeline to keep your Vale image up-to-date.

Where to start?

▓▓ Improving existing content

- Peck away at enforcing style guidelines by picking a particular style guideline and then running the Vale prose linter against your documentation assets.
- "Fix" documentation by correcting Vale-reported errors or warnings related to the style guideline rule.

▓▓ Improving existing style rules

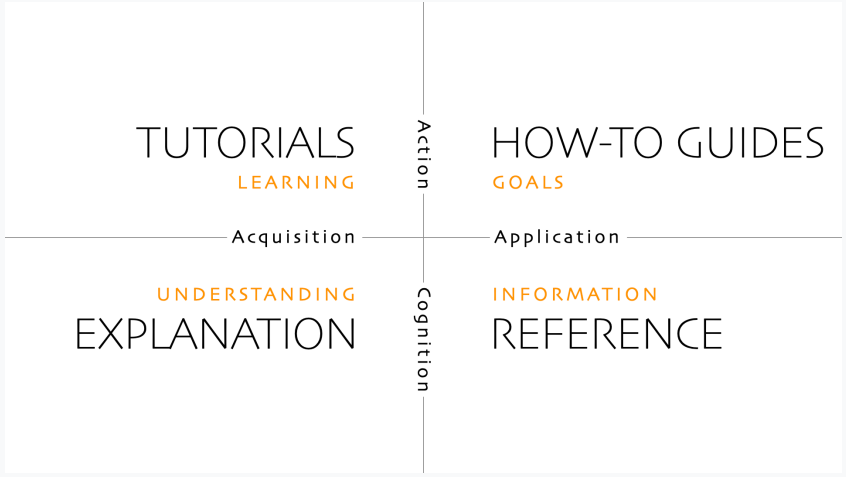
As you use someone else's style guidelines and notice ways to improve their Vale implementation, open issues in their project. Fix them and submit pull requests to improve the project.

▓▓ Creating new style rules

- Create a glossary of terms, specific to your organization.
- Create new prose linter rules to catch issues with documentation to bring consistency to your documentation, based on your glossary.

Where to go after you've started?

Prose linting can only get you so far.



▓▓ Improving documentation structure

- Focus on user needs. Align documentation within a system that promotes usability, for example:
  - Diataxis: Tutorials, how-to guides, explanation, reference
  - DITA: Task, concept, reference, glossary entry, and troubleshooting

▓▓ Improving workflows for creating documentation

- **Red Hat modular documentation system** (*https://redhat-documentation.github.io/modular-docs/*)
  - Helpful resource if you might be looking for templates

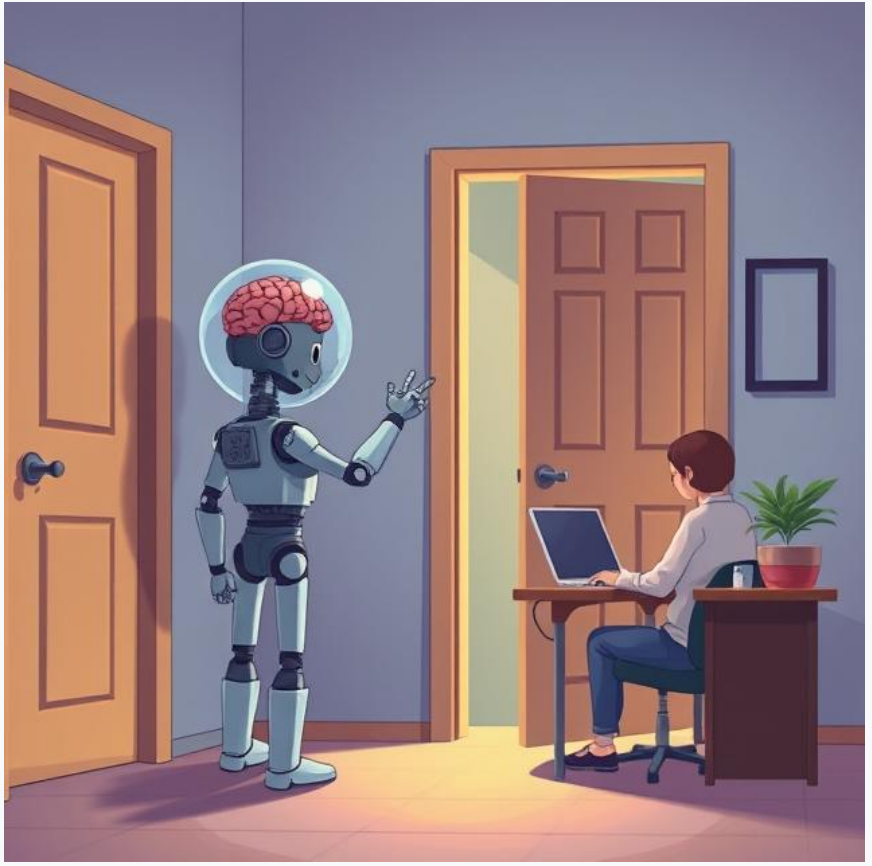
▒▒ Getting started

Example tasks:

- Remove explanations from tutorials and guides.
- Identify repetitive and reusable content in documentation, pull it out and make it a reusable module instead.

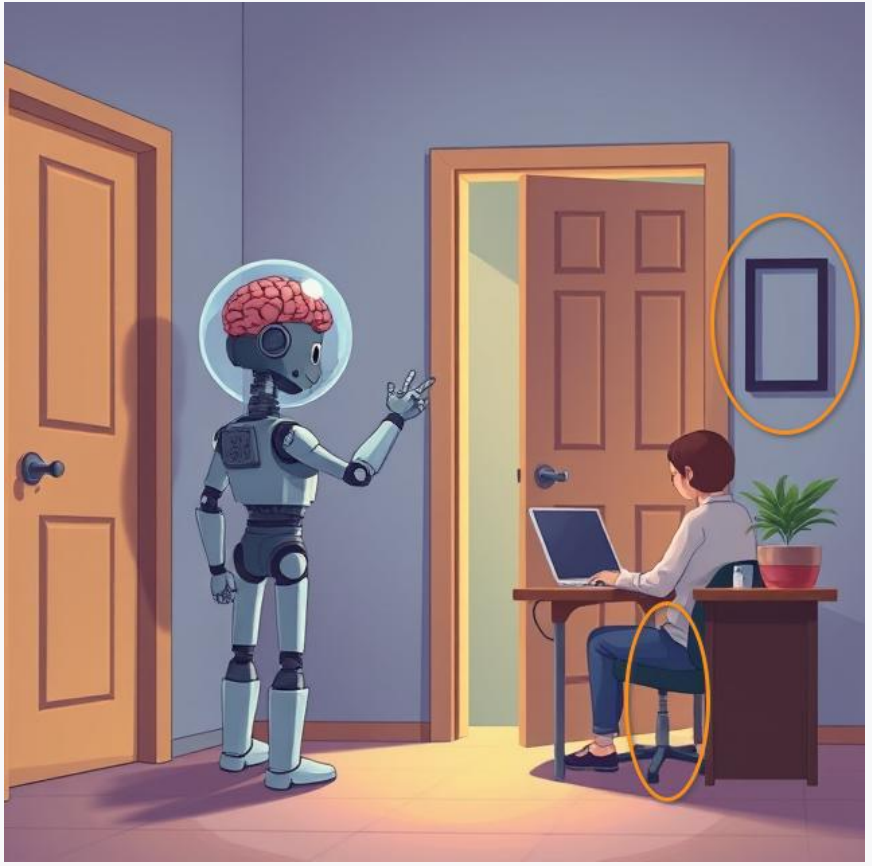
Wait... did you forget something?

> A cartoon image of a robot (whose head is a brain in a jar), knocking its hand on the closed door of a house. Inside the house, there is a person at a desk, typing on a laptop computer. The person's back is turned to the robot at the door.



Wait... did you forget something?

> A cartoon image of a robot (whose head is a brain in a jar), knocking its hand on the closed door of a house. Inside the house, there is a person at a desk, typing on a laptop computer. The person's back is turned to the robot at the door.



Acknowledgements

▓▓ GIMP

> The Free & Open Source Image Editor
>
> — https://www.gimp.org/

▓▓ ImageMagick

> ImageMagick® is a free and open-source software suite, used for editing and manipulating digital images. It can be used to create, edit, compose, or convert bitmap images, and supports a wide range of file formats, including JPEG, PNG, GIF, TIFF, and PDF.
>
> — https://github.com/ImageMagick/ImageMagick

▓▓ Mermaid

> Mermaid is a JavaScript-based diagramming and charting tool that uses Markdown-inspired text definitions and a renderer to create and modify complex diagrams. The main purpose of Mermaid is to help documentation catch up with development.
>
> — https://github.com/mermaid-js/mermaid

▓▓ presenterm

> presenterm lets you create presentations in markdown format and run them from your terminal, with support for image and animated gifs, highly customizable themes, code highlighting, exporting presentations into PDF [and HTML] format, and plenty of other features.
>
> — https://github.com/mfontanini/presenterm

▓▓ qrencode

> Encode input data in a QR Code and save as a PNG or EPS image.
>
> — https://github.com/fukuchi/libqrencode/

The end