Collaboration & Teamwork

The Fundamentals of Tech Writing | BUT 2025

The Red Hat Customer Content Services team

- Workshop: Sample documentation workflow scenario
- Teamwork tools
  - Google docs
  - GitHub
- Collaboration through version control
  - Docs as code
  - Using GitHub web UI
  - Creating Git branches
  - Creating and reviewing pull requests

Workshop: Sample documentation workflow scenarios

AKA time to put your docs hard skills & soft skills to use!

Rules of the workshop

The scenario: 5 writing teams working on separate documentation tickets, and collaborating with their stakeholders.

The goal: Go through the phases of working on a piece of software documentation in a larger team, and overcome the various complications that may occur.

The follow-up: Share with the other team how you approached your task, what challenges you faced, and how you overcame them.

The purpose: Try out the docs creation process, learn what issues may arise, and hopefully have some fun in the process :-)

The stakeholders

- **Pam**, the Product Manager
- **Dave**, the Software Engineer (developer)
- **Quentin**, the Quality Engineer (tester)
- **Cassie,** the customer support specialist
- **Pierre,** a senior Technical Writer on your team
- **Manfred**, a high-ranking tech writing manager

Task 1

"Virt-manager is finally introducing full libvirt support for external snapshots in RHEL 7. This was a customer request, so we need to document it."

(Filed by Dave)

Task 2

Cause: The insights-client service requires permissions to allow execution from cloud-init which were not in the previous selinux-policy versions.
Consequence: Running "insights-client --register" from a cloud-init script fails with several AVCs.
Fix:
Result: Running "insights-client --register" from a cloud-init script does not fail.

(Filed by Quentin)

Task 3

"My org director just sent a memo that in a satisfaction survey, a key customer complained that our docs are lacking info on creating a user in the product GUI. We really should get on top of that ASAP!"

(email by Manfred)

Version control in practice

Collaboration in Google Docs

Task:

Work as an entire class on documenting the use case in this document

- For communication use only **comments** and **suggests** in the doc, or the **#classwork** channel on Discord

WYSIWYG

- Text processor (MS Word, LibreOffice Writer, Google Docs…)
- Granular control of all visual properties of all content elements
- Difficult to standardize, export, and version control
- Best for smaller doc sets with limited contributors

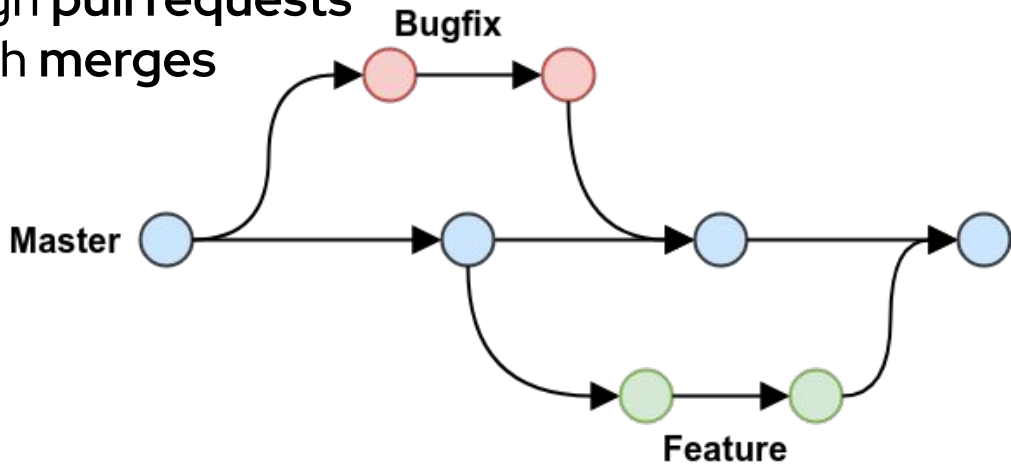
Docs as code

- Markup language (AsciiDoc, DocBook XML, Markdown…) + Text editor/IDE (Vim, VS Code, IntelliJ…)
- Determines types of content elements -> Visual properties rendered through stylesheets
- Easier standardization, exporting, and version control
- Best for larger doc sets with more contributors

Git

For granular version control of software code (and docs as code)

- Code for projects resides in **repositories**
- Changes to code through **commits**
- Applying sets of changes through **branches**
- Review of branches through **pull requests**
- Combine branches through **merges**
- ...and much more



Collaboration in Git - part 1

- **Task:** Add ***a page about yourself*** to the wiki section of the course GitHub repository
  - Share anything you feel like sharing (hobbies, favourite band, etc.)
  - No need to use real names or biographic data - keep in mind the repo is public
  - Use the GitHub web GUI.
- Time allotment: ~15 mins

Collaboration in Git - part 2

- **Task:** Create a documentation module with the recipe for your favourite food
  - Create a new file for your content in the repo
  - Create a new branch for your work (will be requested in the GitHub GUI)
  - Use markdown (Add "*.md*" at the end of the file name)
  - Adhere to technical writing style as much as possible
  - Segment your updates through commits with meaningful messages
- Create a pull request from your branch
- Assign yourself as assignee to the pull request

Collaboration in Git - part 3

- **Task:** Peer review the content

- Assign yourself as reviewer to one of the open pull requests in the repo
  - Add at least 5 comments or suggestions for improvement
  - Use both positive and negative (but constructive) feedback

Collaboration in Git - part 4

Homework!

- **Task:** Apply the peer review
- Respond to the comments and suggestions in your pull request
- Apply the change, or provide reasons why not to apply them
- Mark the pull request as ready (via a comment)

- Optional: In the pull request you peer reviewed, respond to the owner's comments

Thank you