CLARIAH project management via Git and GitHub

Maarten van Gompel

Introduction

In this tutorial:

- Short intro about git and github
- Motivation
- How CLARIAH uses git & github:
  - The CLARIAH-plus repository

Background knowledge

- **Git**: a *version control system*; used by developers for source code management
  - Enables collaboratively working on the same code-base
  - Systematically tracks all edits across time
  - Branching and merging
  - Excels at plain-text content (code, markdown, TeX, HTML, etc)
- **GitHub**: A web-platform with git at its core, and lots of features around it:
  - Issue tracker
  - Pull Request aka Merge Request
  - Projects (kanban board)
  - Continuous Integration & Deployment (GitHub Actions)
  - Teams

Motivation (1)

If we all use the same platform we have:

- Everything in one place
- Full transparency, public accessibility
- Developer-friendly
  - Easy cross-links between development work and project management
- Advanced collaboration mechanism (also with external parties)
- Easy interaction with other projects

Motivation (2)

When **NOT** to use Git/GitHub:

- **Privacy sensitive information!**
- Large data; significant binary data
- Real-time collaborative editing
  - Using MarkDown? Consider HedgeDoc: https://pad.nixnet.services/

CLARIAH on GitHub

CLARIAH 'organization' on github:

`https://github.com/CLARIAH/`

- Namespace holding various CLARIAH-related repositories
- Available to all CLARIAH projects
- Does not necessarily hold all CLARIAH software
- Software may also be stored with other 'organizations' or individual users
  - https://github.com//
  - https://github.com/knaw-huc/textrepo
  - https://github.com/proycon/FoLiA

CLARIAH-plus repository

A single git repository for CLARIAH-plus project management:

`https://github.com/CLARIAH/clariah-plus`

Inside the git repository:

- Storing documents, presentations, notes, minutes related to the CLARIAH-plus project as a whole.
- `interest-groups/`: Input/output of interest groups
  - subdirectory per interest group
  - technical specifications, notes/minutes, presentations
  - ex. software/infrastructure requirements
- `technical-committee/`: Input/output of technical committee as a whole
  - `shared-development-roadmap/` - Shared Development Roadmap v2
- `use-cases/`: Use cases

Contributing files

1. Use the Github web interface
   - Navigate to any file or click `add file` (top right)
   - Press the pencil icon (top right) to change the contents of a file in the browser
   - Benefit from built-in preview facilities for e.g. Markdown
   - Enter a small *commit message* summarizing your changes
   - Press the green commit button
2. Use git from command-line
   - `git clone https://github.com/CLARIAH/clariah-plus/`
   - Use your favourite text editor to edit
   - `git add <file>`
   - `git commit -a`
   - `git push`
3. Use graphical git clients (e.g. GitHub Desktop)

Read and follow the contribution guidelines:
https://github.com/CLARIAH/clariah-plus/blob/main/CONTRIBUTING.md

MarkDown

Use MarkDown syntax for documents as much as possible

- Simple straightforward mark-up syntax in a plain-text format (works well with git)
- Low learning curve, no special tools needed
- **Interoperability**: Can be converted (e.g to LaTeX,PDF,HTML) for print/display.
- May be insufficient for print-focussed documents (papers) or presentations

Using another format?

- `pdf`: yes, feel free to add a pdf version to the repo (alongside the markdown source)
- `tex`: yes, good for papers
- `ppt`, `odp`: okay, just add it to the repo
- `xlsx`, `ods`: okay, just add it to the repo
- `docx`, `odt`: Are you sure you can't do this in MarkDown? If not, just add it to the repo
- Google Document (live) and similar cloud solutions:
  - either export the document and add it (and remove it from the cloud platform)
  - or add a link to the document in the repo (e.g. in a README.md)

Issue tracker: Structured Discussions (1)

Issues: Structured discussions around well-defined themes.

Issues are associated with a specific github repository. The one at `CLARIAH/clariah-plus` is for all and any meta-issues (project planning, shared services):

* `https://github.com/CLARIAH/clariah-plus/issues`

Issues pertaining to specific software project should go in their respective repos: * Often used for bug tracking & feature requests. * Our use is a bit extended...

Issue tracker: Structured Discussions (2)

Features:

- Issues can be easily hyperlinked (to establish relations between issues)
- Issues allow cross-linking discussions to actual work (from git commits) that emerge from the discussions.
- Issues can be classified/tagged with labels
- Issues can be assigned a milestone
- Issues can have people 'assigned' to them
- **Milestones**:
  - Often corresponds to a stage in software development
  - Groups issues
  - Due date

Make sure to **Watch** the `clariah-plus` github repository!

- Automatic mail notifications
- You can reply to issues via mail too!

Pull Requests

Pull Requests: Offers a mechanism for reviewing/accepting the work of others

- https://github.com/CLARIAH/clariah-plus/pulls
- (aka Merge Requests)
- Discussions similar to issues
- Offers a review mechanism, facilitates decision making process
- Recommended for large or possibly controversial updates where you need additional feedback/review.

Project planning

`https://github.com/orgs/CLARIAH/projects?type=beta`

- Kanban boards/lists to organize issues
- Each 'CLARIAH Shared Service' or 'epic' is a project
  - has its own kanban board
  - not specific to a repository
  - can hold issues stemming from multiple repositories
  - Feel free to add additional projects if needed
- We use the beta version of the new Github projects
  - this associated with the CLARIAH organization (not a particular repo like `clariah-plus`)
- Leaders of the 'epics' are responsible for keeping the kanban board up to date.

Teams

- Simple mechanism to group users
- Team per WP
- Team per shared service/epic.

How to contribute?

- Each repository has a `README.md` describing its contents
- Repositories often have a `CONTRIBUTING.md` describing how to contribute:
- Add/edit documents
- Create and reply to issues
- Use the kanban board

Do's and Don'ts (1)

DO:

- Respond to issues you have an opinion on (don't hold back!)
- Add relevant documents to the repository (don't hold back!)
- Use Markdown, edit it in the way and program that suits you best
- Add your IG/TC presentations to the clariah-plus repository
- Make an issue for every ToDo item
- Add `README.md` documents in directories where it makes sense
- Check and follow contribution guidelines (`CONTRIBUTING.md`)
- Often visit and update the kanban board (Github Projects)
- (gently) poke people (`@username`) in issues/PRs when their input is requested

Do's and Don'ts (2)

DON'T:

- Don't add privacy/security sensitive data! (everything is open and public)
- Don't add version information in filenames, git handles versioning for you
- Don't add multiple copies/variants of the same document, git handles branching for you
- Don't be afraid to mess up the git repository, in case of error we can go back easily
- Don't hold 'secret' meetings: either store the minutes of your meeting in the clariah-plus repository or summarize the output in one or more relevant issues.
- Don't use other platforms for todos/project management/issues/document-storage; we keep things in one place for findability and transparency.
- Don't ignore Github mail notifications or let them land in your spam box!