martin-leandro.paleico@gwdg.de

Martin Paleico

git and Gitlab

Collaborative Work, Issue and Feature Tracking

Table of contents

1 Version Control and git

2 Gitlab

Today

- Learn a bit about version control and git (if you haven't already)
- Install Gitlab CE
- Test some of Gitlab's collaborative tools and a bit of git
- Plenary conclusion

Outline

1 Version Control and git

2 Gitlab

Version control for system administrators

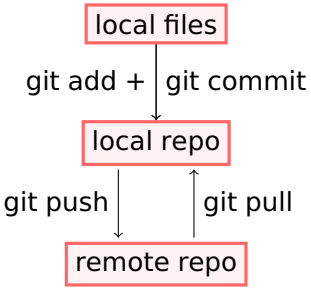
"Framework that allows for keeping track of changes made to files"

- Normally used for code, but relevant for sys. admins: configuration files, documentation, many libraries and services available as repositories
- Versioning
- "Backuping"
- History of changes and reasoning behind them -> Sneaky documentation
- Branching: Work on new features without overwriting base configuration
- Transferrability to other systems (e.g. from dev to live system)
- Collaborative and simultaneous work with other admins

Features of git

- "distributed": no unique central repository for files, many local and remote repositories possible with more or less equal rank
- "non-locking": multiple people can work on the same file (have to deal with it afterwards)
- Many other frameworks (mercurial, subversion, etc) with different philosophies
- Many possibilities for remote repository. Here: Gitlab

git scheme (reminder)



Reality can be much more complicated, see for example
https://blog.osteele.com/2008/05/my-git-workflow/

Good practices when working with git

- Files are stored whole, not as diffs, so don't commit large binaries, images, pdf's/MS Office formats, etc. -> Store the scripts that generate those when possible
- `git-lfs` might help with large files
- Small changes with continuous commits that fix one issue
- Commit functional configurations and code
- Use branches for testing and development
- Mistakes are fixable but sometimes disentangling a repository is hard

Good practices when working with git

- Once something enters a repository, it can be very hard to get rid of it: Careful with passwords, API tokens, etc.!

Docker config file

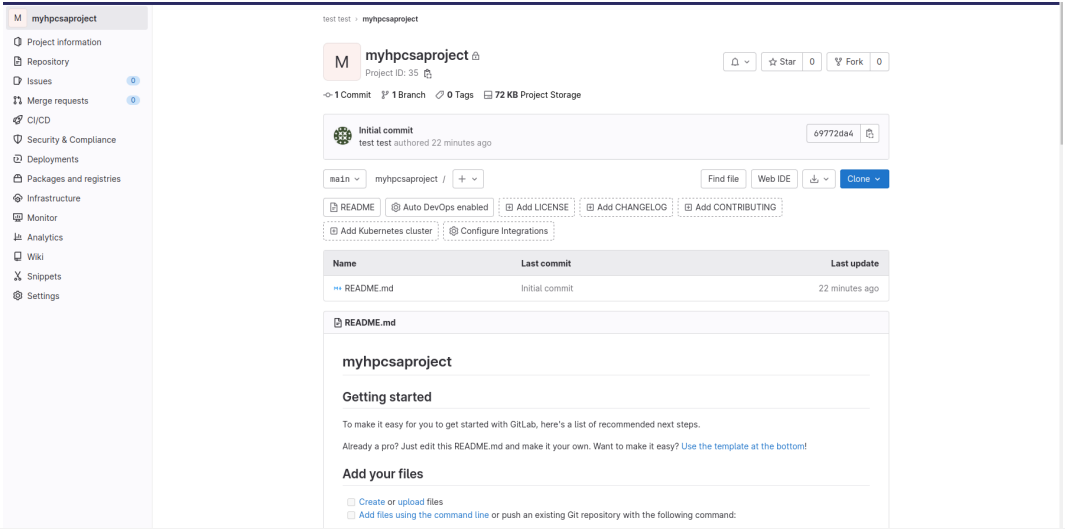
```
version: "3.7"
services:
  omeroserver:
    image: "omero-server-with-figure:5.6.5"
    user: 'omero-server'
    environment:
      CONFIG_omero_db_host: database
      CONFIG_omero_db_user: omero
      CONFIG_omero_db_pass: omero
      CONFIG_omero_db_name: omero
      ROOTPASS: OoPsPl41n73x7p4s5w0rd
```

Outline

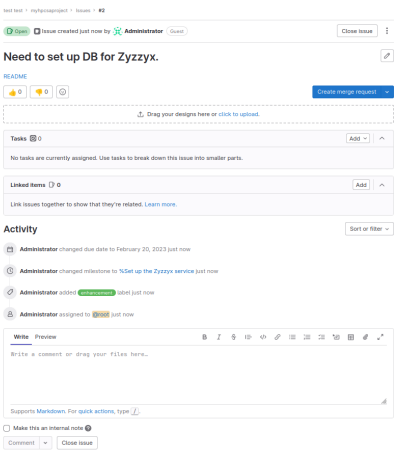
1 Version Control and git

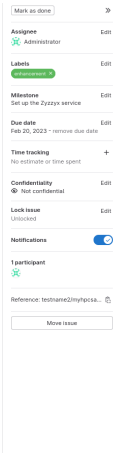
2 Gitlab

Collaborative Work: Gitlab



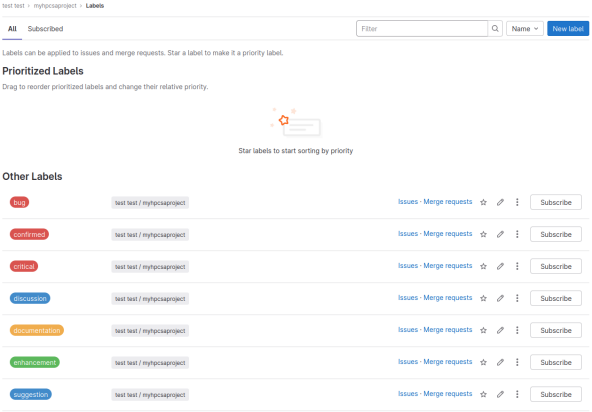
Collaborative Work: Issues





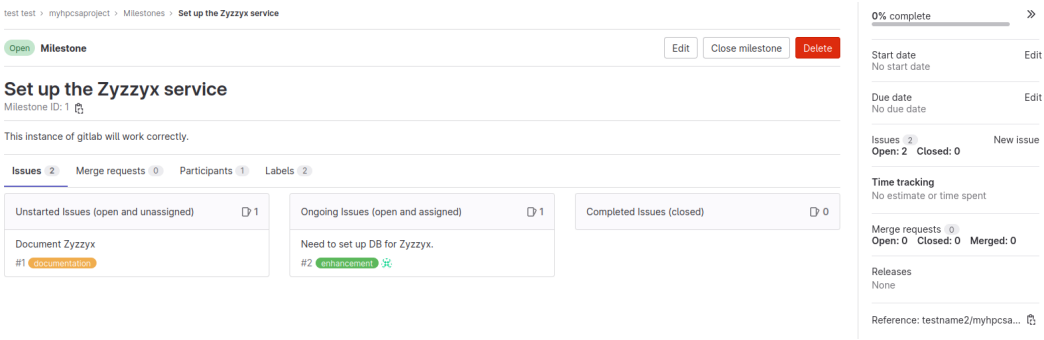
- An issue can be a problem with one of your services
- An issue can also be a task that you need to achieve, such as a new service or feature for an existing service
- This could be use to track and have a record of your own tasks and work

Collaborative Work: Labels



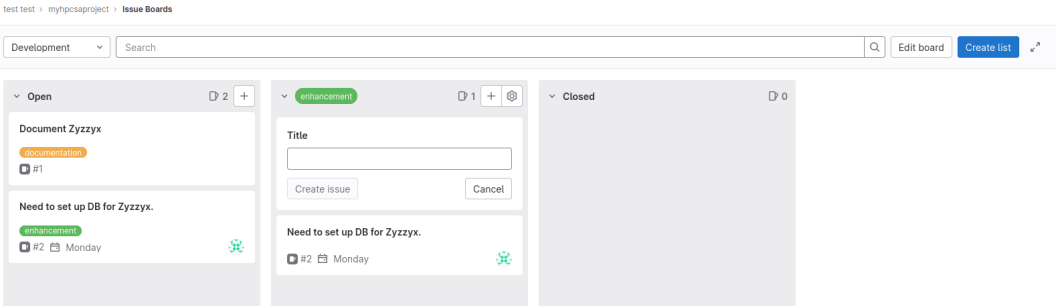
- Labels help you categorize issues
- You can subscribe to labels and get notifications, or use them to create boards

Collaborative Work: Milestones



- Collect tasks to achieve a specific goal
- Track progress of your goals

Collaborative Work: Boards



- Visualize issues at a glance

Plenary Discussion

- Have you used version control before? Have you used feature and issue tracking?
- Would you use Gitlab's tools? Any other feature you would need?
- Any other tools that you use for this sort of work?
- Possible security concerns? Reverse-proxying? Containerization?
- curl plus sudo bash on setup, dangerous
- "Impersonate User": Common feature, good for debugging, security and data concerns
- Compare features with ticketing system exercise
- You probably want to deactivate the service sine it consumes quite a bit of resources (see exercise sheet for instructions)
- Look-back at previous "best practices" presentation

Other Links and Resources

- Hints for good commit messages: `https://cbea.ms/git-commit/`
- Making your own git in Python: `https://www.leshenko.net/p/ugit`