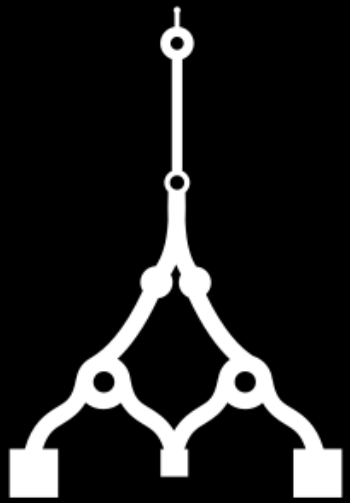

MERCURIAL

CONFERENCE - PARIS

April 05-07, 2023

17 years of Mercurial at Logilab

… and 23 years of version control !

TLDR: Timeline of VCS

Version Control Systems have been used to:

1. record the changes of a file (versions)
2. record the changes of a set of files (commits)
3. track the process of review and collaboration (branches)
4. support changing the changes (evolution of draft commits)

Before it started - command `cp`

No tool

`cp -r` of files and directories

name them with date or number

use `diff`, `mv` or `ln -s` to go back in time

Before I started - RCS

Revision Control System

`rcs` command to go back in time

record multiple revisions of a single file

no network features

2000 - CVS

Concurrent Version System

`cvs` to exchange files with centralized network server

revisions handled file by file

no branches

2005 - Subversion

centralized network server

commits with multiple files

branches

branching and merging not easy

2005/2006 - So many options !

Linux Kernel stopped using BitKeeper because of licensing issues: many tools appeared !

We tested mercurial, git, darcs, etc. then picked hg because better CLI and extendable since written in Python

We switched everything to mercurial within a few months.

2006 - Mercurial

decentralized: `ci/up` are local then `push/pull`

no server but shared repositories via ssh accounts

Mercurial-Server to have r/w permissions and http urls

Contribute to Mercurial

Free software written in Python with an extension system

Contributed to extensions

Hosted development sprints

Review process

Collaborative software development requires review

`mq` extension to manage patches to apply as with `quilt`

create a companion repository to version the patches, then review and apply

Iterative reviews

Review requires back-and-forth communication, but with `mq` difficult to rework a stack of patches

need to push/pull/rebase patches as easily as commits

P.Y. David employed at Logilab. Got hooked to Mercurial and started `phase` then `evolve` before leaving to Facebook.

Forges to support the software dev. process

Rise of GitHub, GitLab and others

Logilab included Mercurial-Server and draft changesets in the forge it was developing (projects, issues, versions, CI, artifact repositories, etc)

2020 - Heptapod

Logilab's own forge abandoned, switch to heptapod with support from Octobus

Everything is version controlled

in 2023 as in 2000, we put everything under version control:

internal documentation, commercial offers, accounting,

source code, infrastructure code, etc.

Intensive use of `.gitlab-ci.yml`

jobs to lint, test, build doc and containers, deploy in kubernetes with helm, update infra, etc.

scheduled or manual pipelines to generate documents, update or sync data, etc.

Pages and artifact repositories

manuals, reports, dashboards and websites using `pages`

Python packages, Containers, Helm Charts, etc.

On-demand applications

start and stop rewiev apps

CubicWeb-as-a-Service apps built from ontologies

With API Almost a `Function-as-a-Service` platform

What else ?

no code search (tried sourcegraph)

`code-doctor` to generate Merge Requests to upgrade

python, ci, dependencies, etc.

use LLM maybe ?

Version Control Is Everywhere

the idea of version control has spread everywhere:

infrastructure (gitops), datasets, work environments: tools can create versions and push/pull

Want more ?

Law as Code: talk at conference "La Fabrique de la loi" in 2012 => law process is like version control + review process

Training: use commits to drive an exercise step by step.

Mercurial is at the heart of Logilab


mercurial

Contact

Nicolas.Chauvat@logilab.fr