Jujutsu – Die sanfte Technik der Versionskontrolle

Leah Neukirchen

<leah@vuxu.org>

ASM'25, München

2025-05-31

Worum geht's?

- Jujutsu (`jj`) ist ein Versionskontrollsystem das von Martin von Zweigbergk und Mitstreitenden seit 2019 entwickelt wird.
- In Rust geschrieben
- Modulares Design
- Verschiedene Backends
  - Git (auch simultan zu normalen Git-Befehlen: *colocation*)

Ikonoklastik

Fundamentale Unterschiede zu Git:

- anonyme Branches
- Arbeitsverzeichnis als Commit (keine Staging-Area)
- Konflikte als Objekte erster Klasse
- automatisches Rebasing
- Changes haben Geschichte
- Weniger Konzepte → einfacher zu lernen(?)

Commit ID und Change ID

Git identifiziert jeden Commit eindeutig[1] durch seinen Inhalt. Wenn man `rebase`/`commit --amend`/… verwendet, ändern sich Commit IDs!

Jujutsu verwendet das gleiche Datenmodell, aber jeder Commit hat auch noch eine (zufällig generierte) *Change ID*, die bei solchen Operationen erhalten bleibt.

```
% jj status
Working copy changes:
M README
Working copy  (@) : lxyqoypw 1d3f2732 Adjust README
Parent commit (@-): plzqusrz b01a2bf2 Adjust README
```

Commits können als unveränderlich markiert werden (z. B. gepushte Commits).

[1] bis auf Hash-Kollisionen

Aber wie arbeitet man denn so?

- `jj new`, `jj edit`
- `jj describe`
- `jj status`
- `jj split`, `jj squash`, `jj rebase`

Demonstration

Das OpLog

Alle Zustände des Repositories werden in einem Log gespeichert und sind wiederherstellbar.

- `jj op log`, `jj op show`
- `jj op restore`
- Bequemer: `jj undo`

Die RevSet-Sprache

Ähnlich zu Mercurial gibt es eine kleine DSL, um Commits zu spezifizieren.

Im Folgenden ist x ein Tag, Bookmark, Git ref, Commit ID oder Change ID.

`x-`	Eltern von x
`x+`	Kinder von x
`x::y`	Abkömlinge von x die Vorfahren von y sind
`x..y`	Vorfahren von y die nicht Vorfahren von y sind
`x`	Commits nicht in x
`x&y`	Commits in x und y
`x~y`	Commits in x ohne Commits in y
`x|y`	Commits in x oder y

Workflows

Zwei gängige Ansätze:

squash-Workflow, ähnlich zu Git

```
jj describe -m '...'
jj new
jj squash
```

`edit`-Workflow

```
jj new -m '...'
jj edit
```

Git Interop

```
jj git fetch
jj git push
jj git push -c @

jj bookmark set push-vmunwxsksqvk
jj git push
```

Ecken und Enden

- Forges können nur Git
  - Schwer, die `jj`-Vorteile kollektiv einzusetzen
  - Aber: neuer Header „Change-ID“ kommt
- Kein Magit! :(
  - Sonstiger IDE-Support auch noch nicht optimal
- Performance-Probleme auf alten Maschinen

Die Entwicklung dauert an und es gibt monatlich neue Features!

Vielen Dank!

Noch Fragen?