Jujutsu

David Darras,Samuel Lepin

19 novembre 2025

Résumé

Jujutsu (jj) est un système de contrôle de version moderne conçu pour simplifier les workflows et améliorer la sécurité par rapport à Git. Il supprime la staging area, traite le répertoire de travail comme un commit actif et introduit les *changes*, des unités de travail modifiables et traçables. Grâce à son *operation log*, toutes les actions sont réversibles. Écrit en Rust et compatible avec Git, jj propose une expérience plus intuitive et permet d'expérimenter sans risque. Cette approche en fait une alternative prometteuse pour les projets personnels mais difficile à imposer sur un marché saturé par Git.

Abstract

Jujutsu (jj) is a modern distributed version control system designed to provide a simpler and safer workflow compared to Git. It removes the staging area, treats the working directory as an active commit, and introduces mutable *changes* that preserve logical traceability. Through its global *operation log*, every action can be undone safely, encouraging experimentation without data loss. Written in Rust and fully compatible with Git, jj offers a more intuitive user experience while maintaining strong performance. This approach makes it a promising alternative for personal projects, but difficult to impose on a market saturated by Git.

Keywords: Version Control System, Git, Jujutsu

Introduction

Les systèmes de contrôle de version (*Version Control System – VCS*) jouent un rôle central dans le développement logiciel moderne. Ils permettent de coordonner efficacement le travail d'équipes de développeurs tout en conservant un historique complet des modifications apportées au code, ce qui réduit considérablement les conflits et le risque de pertes accidentelles de données. Grâce à un VCS, il est possible de revenir à des versions antérieures des fichiers ou des branches en cas d'erreur, tandis que l'usage de branches facilite l'ajout de nouvelles fonctionnalités sans perturber le code stable. Chaque changement est documenté par des commits, assurant la traçabilité des modifications et la possibilité d'identifier qui a effectué quoi et quand. Ces capacités font des VCS des outils indispensables pour gérer la complexité croissante des projets logiciels et maintenir la qualité du code.

Selon une enquête publiée en 2023 sur le *Stack Overflow Blog* [3], **Git** domine largement le paysage des systèmes de contrôle de version avec une popularité de 93,78 %. Les systèmes historiques comme **Subversion** (SVN) et **Mercurial** sont désormais minoritaires, avec respectivement 5,18 % et 1,13 % de popularité. L'écosystème Git est omniprésent, notamment avec des plateformes telles que **GitHub** et **GitLab**. Cependant, Git présente certaines limites : la complexité de ses commandes, la présence obligatoire d'une *staging area* et la difficulté de modifier l'historique des commits passés représentent des contraintes pour les développeurs.

Présentation générale de Jujutsu

Jujutsu (ou `jj`) est un système de contrôle de version distribué open source relativement récent, créé par Martin von Zweigbergk, ingénieur chez Google, en 2019 [1]. Écrit en Rust et publié sous licence Apache 2.0, Jujutsu vise à offrir une alternative plus simple et plus performante à Git, tout en restant **compatible avec l'écosystème Git** existant. L'objectif est de réduire le nombre de concepts que l'utilisateur doit maîtriser et de rendre toutes les opérations réversibles par défaut avec `jj undo`, permettant un travail plus sûr et plus flexible.

Un workflow repensé

Dans Jujutsu, le répertoire de travail, appelé **copy directory**, est traité comme un commit actif. Toutes les modifications apportées aux fichiers sont automatiquement prises en compte, ce qui élimine le besoin de les préparer manuellement avant validation. Cette approche simplifie le suivi des modifications et réduit les risques liés à un oubli de `git add`.

Jujutsu introduit le concept de *change*, qui représente une unité de travail modifiable et distincte du commit classique. Chaque change correspond à une idée ou à une tâche et peut être ajusté après sa création sans altérer l'historique. Lorsqu'un commit est généré à partir d'un change, un hash immuable est calculé pour le commit, mais le *ChangeID* reste identique, même si le commit subit des modifications ou un rebase. Cette séparation entre ChangeID et commit hash permet de retravailler, corriger ou compléter un changement tout en conservant sa traçabilité logique. De plus, après la création d'un commit, il est possible de modifier le message du commit, les fichiers inclus ou la description du change, offrant une flexibilité que Git ne permet qu'au prix d'opérations complexes.

Bookmarks : une alternative aux branches

Dans Jujutsu, les branches traditionnelles de Git sont remplacées par les *bookmarks*. Un bookmark est une étiquette qui pointe vers un change spécifique et sert à suivre l'évolution d'une fonctionnalité particulière. Les bookmarks permettent de renommer ou de déplacer une référence sans affecter l'historique des changes existants.

Chaque utilisateur dispose de sa propre copie locale de Jujutsu. Une personne peut donc travailler de manière transparente sur ses bookmarks et ses changes sans que les autres utilisateurs soient impactés. Lorsqu'un bookmark est lié à un dépôt distant et qu'un commit est poussé, ce commit et ses descendants deviennent immuables, garantissant l'intégrité de l'historique et facilitant la collaboration avec les remotes Git.

Fonctionnalités avancées de Jujutsu

Jujutsu repose également sur plusieurs mécanismes, parmi eux, l'**operation log** occupe une place centrale. Contrairement au reflog de Git, qui enregistre uniquement les déplacements de références, l'operation log consigne toutes les actions réalisées par l'utilisateur : création de change, modification, rebase, merge, suppression, etc. Grâce à cet historique complet, il est possible d'annuler *n'importe quelle* opération via `jj undo`, transformant la gestion des erreurs en un processus simple et fiable. Cette approche réduit considérablement les risques d'erreurs irréversibles et encourage l'expérimentation et l'erreur dans le développement.

Jujutsu innove également dans la gestion des conflits. Dans Git, un conflit doit être résolu immédiatement pour pouvoir poursuivre le workflow. Dans jj, les **conflits commitables** constituent un changement à part entière. Ils peuvent être enregistrés, partagés ou modifiés ultérieurement, ce qui libère l'utilisateur de la pression de résoudre le conflit instan-

tanément. Cette approche est particulièrement utile dans des projets complexes où les merges sont fréquents.

Un autre point fort est le **rebase automatique**. Lorsqu'un change est modifié, Jujutsu met automatiquement à jour tous ses descendants. Cela signifie qu'il n'est plus nécessaire d'effectuer manuellement des rebases successifs pour maintenir un historique cohérent. Ce comportement réduit le risque d'erreur, limite la duplication de commits et facilite la gestion d'historiques longs et ramifiés.

Comparé à Git, Jujutsu propose donc une expérience plus fluide. Git repose sur des commits immuables, une staging area souvent coûteuse en erreurs, et une gestion du reflog peu intuitive. Jujutsu supprime ces obstacles. Les changes sont mutables, l'historique est malléable, les erreurs sont récupérables et la staging area disparaît. Cependant, malgré ces atouts, certaines limites demeurent. Jujutsu dispose d'une communauté plus restreinte, d'une documentation encore en développement, et d'un écosystème plus jeune. Les intégrations dans les IDE restent limitées par rapport à Git, tout comme les outils graphiques.

Conclusion

Jujutsu propose une approche simplifiée et flexible du contrôle de version, adaptée aux projets où la complexité des workflows Git n'est pas nécessaire. Son système de changes modifiables, la gestion automatique du répertoire de travail et les bookmarks permettent un développement itératif plus sûr et plus fluide. Bien que certaines fonctionnalités avancées de Git ne soient pas encore pleinement prises en charge, Jujutsu reste un outil prometteur pour les équipes petites ou expérimentales et les nouveaux projets.

Références commentées

- **Alden et al. (2024).[1]** Article détaillant la conception de Jujutsu, ses principes fondamentaux ainsi que ses différences structurantes avec Git.
- **Donovan (2023), Stack Overflow Developer Survey.[3]** Cette enquête fournit les statistiques essentielles sur la domination de Git dans l'industrie.
- **Documentation officielle de Jujutsu.[2]** Cette documentation décrit les commandes, les concepts clés (changes, revsets, bookmarks) et le fonctionnement du *operation log*.

References

[1] Daroc Alden. Jujutsu: a new, git-compatible version control system, 2024. URL: `https://lwn.net/Articles/958468/`.

[2] Jujutsu Contributors. Jujutsu: A version control system — project readme, 2025. Accessed: 2025-11-24. URL: `https://github.com/jj-vcs/jj`.

[3] Ryan Donovan. Beyond git: The other version control systems developers use, 2023. URL: `https://stackoverflow.blog/2023/01/09/beyond-git-the-other-version-control-systems-developers-use/`.