Jujutsu Git-compatible version control

CONCEPTS

There is no staging area: the working copy is itself a commit (@), and file edits amend it automatically.

Edit files amends @
↓ `jj describe`
Describe set the message
↓ `jj new`
New change fresh empty @
↓ `jj bookmark set`
Bookmark branch pointer
↓ `jj git push`
Remote origin

Change IDs (letters) stay stable when a change is rewritten; commit IDs (hex) change. Use change IDs.

Bookmarks do not follow @ automatically: move them with jj bookmark set before pushing.

SETUP

`jj git clone` *`<url>`*
Clone a Git repository

`jj git init --colocate`
Use jj in an existing Git repo

`jj config set --user user.name "`*`<name>`***`"`**
Set your name

CHANGES

`jj st`
Show the working-copy change

`jj diff`
Diff of the current change

`jj show` *`<rev>`*
Message and diff of a change

`jj describe -m "`*`<msg>`***`"`**
Set the current change's message

`jj commit -m "`*`<msg>`***`"`**
Describe @ and start a new one

CHANGES (cont.)

`jj new`
Start a new empty change on @

`jj new` *`<rev>`*
Start a new change on <rev>

`jj edit` *`<rev>`*
Make <rev> the working copy

`jj restore` *`<path>`*
Discard changes to a file

HISTORY

`jj log`
Show the revision graph

`jj log -r` *`<revset>`*
Log selected revisions

REWRITING

`jj squash`
Move @ into its parent

`jj squash --into` *`<rev>`*
Move @ into any ancestor

`jj split`
Split @ into two changes

`jj absorb`
Auto-squash edits into ancestors

`jj rebase -s` *`<rev>`* **`-o`** *`<dest>`*
Move a change and its descendants

`jj abandon` *`<rev>`*
Drop a change, keep its children

BOOKMARKS & REMOTES

`jj bookmark set` *`<name>`* **`-r`** *`<rev>`*
Point a bookmark at a revision

`jj git fetch`
Fetch from the remote

`jj git push`
Push changed bookmarks

UNDO & CONFLICTS

`jj undo`
Undo the last operation