Jujutsu Git-compatible version control

START HERE

`jj git clone <url>`
Copy a repository

`jj git push`
Share bookmarks

`jj st`
What changed?

`jj undo`
Revert the last operation

`jj describe -m "<msg>"`
Name the current change

`jj new`
Start the next change

`jj log`
See history

`jj squash`
Fold into the parent

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

CONCEPTS

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

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.

Rebasing rewrites descendants automatically and conflicts are stored in commits, so nothing blocks you.

SETUP

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

`jj git init --colocate`
Use jj in an existing Git repo
Keeps a .git directory so git and other Git tools keep working.

`jj git init`
Create a new repository here

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

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

SETUP (cont.)

`jj config edit --user`
Edit the user config file

`jj config list`
Show active settings

CHANGES

`jj st`
Show the working-copy change

`jj diff`
Diff of the current change

`jj diff -r <rev>`
Diff of another change

`jj diff --from <a> --to <b>`
Compare two revisions

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

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

`jj describe <rev>`
Edit another change's message

`jj commit -m "<msg>"`
Describe @ and start a new one
Shortcut for describe followed by new.

`jj new`
Start a new empty change on @

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

CHANGES (cont.)

`jj new -m "<msg>"`
New change with a message

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

`jj next / jj prev`
Step to child or parent

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

`jj file untrack <path>`
Stop tracking a file

HISTORY

`jj log`
Show the revision graph

`jj log -r <revset>`
Log selected revisions
Examples: @, @-, trunk()..@, mine().

`jj log -p`
Log with patches

`jj log -n <n> -G`
Last n changes, no graph

`jj evolog`
How a change evolved

`jj file annotate <path>`
Blame a file

`jj file show -r <rev> <path>`
Print a file at a revision

`jj file list`
List tracked files

REWRITING

`jj squash`
Move @ into its parent

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

`jj squash -i`
Pick which parts to move

`jj split`
Split @ into two changes

`jj absorb`
Auto-squash edits into ancestors
Each hunk goes to the mutable ancestor that last touched those lines.

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

`jj rebase -r <rev> -o <dest>`
Move only one change

`jj rebase -b <rev> -o <dest>`
Move a whole branch

`jj duplicate <rev>`
Copy a change

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

`jj revert -r <rev> -o @`
Apply the inverse of a change

`jj diffedit -r <rev>`
Edit a change's diff in a tool

BOOKMARKS & REMOTES

`jj bookmark list`
List bookmarks

`jj bookmark create <name>`
Create a bookmark at @

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

`jj bookmark delete <name>`
Delete a bookmark

`jj bookmark track <name>@<remote>`
Track a remote bookmark

`jj git fetch`
Fetch from the remote

`jj git push`
Push changed bookmarks

`jj git push -b <name>`
Push one bookmark

`jj git push -c <rev>`
Push with an auto-named bookmark

`jj git push --dry-run`
Preview what would be pushed

`jj git remote add <name> <url>`
Add a remote

`jj git remote list`
List remotes

UNDO & CONFLICTS

`jj undo`
Undo the last operation

UNDO & CONFLICTS (cont.)

`jj redo`
Redo an undone operation

`jj op log`
List past operations

`jj op restore <id>`
Restore the repo to an operation

`jj resolve`
Resolve conflicts with a merge tool

`jj resolve --list`
List conflicted files

RECIPES

Feature change and push

```
jj git fetch
jj new main -m "Add feature"
# edit files
jj bookmark set my-feature -r @
jj git push -b my-feature
```

Fix an earlier change

```
jj new <rev>
# edit files to fix it
jj squash          # fold fix
into the parent
# or: jj absorb
```

Rebase onto updated main

```
jj git fetch
jj rebase -s <rev> -o
main@origin
jj log
```

Recover from a mistake

```
jj op log
jj undo
# or go back further
jj op restore <id>
```