

### Tricks

To see change of last commits in your IDE, cancel last commit with `git reset HEAD~`. Last commit's will be highlighted similarly to change you are making right now.

To work on a Gerrit CL you don't have locally, use Gerrit's download feature, then `git cl issue ClNumber`, so that any change you made is pushed back on the same CL

If you think you may have lost data, do NOT do `rebase -update`, as it triggers a garbage collection

`rm -f .git/index; git reset`` if git output inconsistent information, that recompute the index that is used to detect local change

### git reflog example

After `git commit --amend` I realized I preferred the last version of the branch.  
How do I get it back?

`git reflog myBranch` to find the hash,  
then `git reset --hard HASH`

After `git pull`, I got a merge conflict. I resolved it incorrectly.  
Now, I realise I'd prefer to undo the pull and resolve again.

`git reflog myBranch` to find the hash,  
then `git reset --hard origin /main; git rebase H`  
`ASH`

I used `git commit` but can't find back the branch with my commit.

`git reflog` to find the hash,  
then `git branch recove red Branch HASH`

### Git log examples

`git log -p -1` Show all change in last commit

`git log -G "myFunction(" folder` Show all places `myFunction` was called in `folder`  
Extremely costly if not restricted to some folder

`git log -L 15:25:myFile` If `myFunction` is currently defined in line 15-25 of `myFile`,  
this show all commits changing `myFunction`, including file rename (but not move inside the file)

### Branch edition

<code>git new-branch branch-_name</code>	New branch tracking origin/main
<code>git new-branch branch-_name --upstream_c-urent</code>	New branch tracking current branch
<code>git branch branch_name commitHash; git switch branch_name</code>	New branch starting at commitHash
<code>git branch child --set--upstream-to parent</code>	Ensure <code>child</code> tracks <code>parent</code> with <code>rebase -update</code> and <code>pull</code>
<code>git branch child --set--upstream-to origin/main</code>	Ensure child does not depends on another branch and gets updated with <code>pull/rebase -update</code>

### Information on current branches

<code>git map-branches</code>	The tree
<code>git map-branches -v</code>	with number of commits
<code>git map-branches -vv</code>	and hash and url
<code>git map-branches -vvv</code>	and gerrit status

### Change branch

<code>git switch myBranch</code>	Switch to myBranch
<code>git cl checkout 1234</code>	Switch to branch linked to gerrit CL 1234

### File edition

<code>./tools/boilerplate.py file.{h,mm}</code>	Create files require <code>git add</code>
<code>./tools/git/move_source_file.py old_file.ext new_file.ext</code>	Move file edit imports and guards

### Commit all current changes

<code>git add .</code>	if new/renamed files
<code>git commit -a -m "Commit message"</code>	New commit
<code>git commit -a --amend --no-edit</code>	Edit the commit
<code>git reset HEAD~</code>	Cancel last commit Keep all changes

### Update to latest version of the code

<code>git rebase-update</code>	fetch latest version of chrome code, update all branches that has upstream-set (see <code>map-branch</code> )
<code>git rebase-update -n</code>	Restart rebase-update after a conflict. Does not fetch
<code>git pull origin main</code>	Fetch latest version of chrome code and put your change on top of it
<code>git pull</code>	Apply your code on top of latest version of the tracked branch; (usually origin/main)
<code>git add .;</code> <code>git rebase --continue</code>	To use after conflict resolution

### Pause and restart current changes

<code>git stash</code>	save for later
<code>git stash list</code>	show all stashes
<code>git stash pop</code>	reapply last stash
<code>git stash pop 3</code>	reapply stash 3 from <code>list</code>
<code>git stash show 3</code>	display content of stash 3 Works as <code>git log</code>

### Clean changes

<code>git cl format</code>	Format edited code
<code>./ios/build/tools/update_deps.py out/FOLDER .</code>	Add missing dependencies Not ios specific. Beta

### Gerrit upload

<code>git cl upload</code>	Upload diff between current branch and upstream
<code>-d</code>	start dry-run
<code>-r googler</code>	add <code>googler</code> as reviewer
<code>--r-owners</code>	automatically choose reviewers for all files
<code>-t "patch message"</code>	(not on first upload)



By arthurmilchior

Published 28th June, 2023.

Last updated 29th June, 2023.

Page 2 of 3.

Sponsored by [Readable.com](https://readable.com)

Measure your website readability!

<https://readable.com>

### Gerrit

git cl web	open this cl on gerrit
git cl issue	the gerrit cl number
git cl issue 1234	Link this branch to gerrit cl 1234
git cl lol	

### See local changes

git diff HEAD	Change since last commit
git diff	changes that are not <code>git added</code>
git diff origin/main	Change from origin/main
git diff HEAD~	The commit you'd get with <code>get commit -a --n o-edit</code>

### git log

git log	list current commit and its parents
git log branchName	list last commit of branchName and its parents
git log commitHash	list commit commitHash and its parents
-p	Show the change in each files
-2	Only show two last commits
-L 12:15:file	Restrict to commits touching lines 12 to 15 of <code>file</code>
-G "myRegexp"	Restrict to commits adding a change matching <code>my Regexp</code>
file_1 file_2 ... file_n	restrict to commits editing some of <code>file_i</code>
--author name	restrict to commit authored by <code>name</code> You only need to enter a small part of the name

### reflog

git reflog	List all recent commits. Even those that are not in branches anymore.
git reflog branchName	List all past and present version of branchName

### Working with a commit

git show commitHash	Show the content of <code>commitHash</code>
git show commitHash:file	show <code>file</code> at commit <code>commitHash</code>
git cherry-pick commitHash	Apply <code>commitHash</code> on top of current branch
git restore -s commitHash file	Change <code>file</code> to the version at <code>commitHash</code>

### Navigating commits

commitRef~	The parent of <code>commitRef</code>
commitRef~3	The grand-grand-parent of <code>commitRef</code>
branchName@{yesterday}	The commit of <code>branchName</code> yesterday



By arthurmilchior

Published 28th June, 2023.

Last updated 29th June, 2023.

Page 3 of 3.

Sponsored by [Readable.com](#)

Measure your website readability!

<https://readable.com>