Cheatography

Chrome-git Cheat Sheet

by arthurmilchior via cheatography.com/188825/cs/39386/

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 your 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 prefered 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 HASH`
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

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

Information on current branches

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

Change branch

git switch myBranch	Switch to myBranch
git cl checkout 1234	Switch to branch linked to gerrit CL 1234

File edition

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

Commit all current changes

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

Update to lastest version of the code

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

Pause and restart current changes

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

Clean changes

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

Gerrit upload

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

By **arthurmilchior**

Published 28th June, 2023.
Last updated 29th June, 2023.

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 `git add`ed
git diff origin/main	Change from origin/main
git diff HEAD~	The commit you'd get with `get commit -a --no-edit`

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 `file`
-G "myRegexp"	Restrict to commits adding a change matching `myRegexp`
file_1 file_2 ... file_n	restrict to commits editing some of file_i
--author name	restrict to commit authored by `name` 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 `commitHash`
git show commitHash:file	show `file` at commit commitHash
git cherry-pick commitHash	Apply commitHash on top of current branch
git restore -s commitHash file	Change `file` to the version at commitHash

Navigating commits

commitRef~	The parent of commitRef
commitRef~3	The grand-grand-parent of commitRef
branchName@{yesterday}	The commit of branchName yesterday