hofmannsven / **README.md**

Last active 7 hours ago • Report gist

My simply Git Cheatsheet

README.md

# Using Git

## Global Settings

Related Setup: https://gist.github.com/hofmannsven/6814278

Related Pro Tips: https://ochronus.com/git-tips-from-the-trenches/

Interactive Beginners Tutorial: http://try.github.io/

## Reminder

Press `minus + shift + s` and `return` to chop/fold long lines!

Show folder content: `ls -la`

## Notes

Do not put (external) dependencies in version control!

## Setup

See where Git is located: `which git`

Get the version of Git: `git --version`

Create an alias (shortcut) for `git status` : `git config --global alias.st status`

## Help

Help: `git help`

## General

Initialize Git: `git init`

Get everything ready to commit: `git add .`

Get custom file ready to commit: `git add index.html`

Commit changes: `git commit -m "Message"`

Add and commit in one step: `git commit -am "Message"`

Remove files from Git: `git rm index.html`

Update all changes: `git add -u`

Remove file but do not track anymore: `git rm --cached index.html`

Move or rename files: `git mv index.html dir/index_new.html`

Undo modifications (restore files from latest commited version): `git checkout -- index.html`

Restore file from a custom commit (in current branch): `git checkout 6eb715d -- index.html`

## Reset

Go back to commit: `git revert 073791e7dd71b90daa853b2c5acc2c925f02dbc6`

Soft reset (move HEAD only; neither staging nor working dir is changed): `git reset --soft 073791e7dd71b90daa853b2c5acc2c925f02dbc6`

Undo latest commit: `git reset --soft HEAD~`

Mixed reset (move HEAD and change staging to match repo; does not affect working dir): `git reset --mixed 073791e7dd71b90daa853b2c5acc2c925f02dbc6`

Hard reset (move HEAD and change staging dir and working dir to match repo): `git reset --hard 073791e7dd71b90daa853b2c5acc2c925f02dbc6`

## Update & Delete

Test-Delete untracked files: `git clean -n`

Delete untracked files (not staging): `git clean -f`

Unstage (undo adds): `git reset HEAD index.html`

Commit to most recent commit: `git commit --amend -m "Message"`

Update most recent commit message: `git commit --amend -m "New Message"`

## Branch

Show branches: `git branch`

Create branch: `git branch branchname`

Change to branch: `git checkout branchname`

Create and change to new branch: `git checkout -b branchname`

Rename branch: `git branch -m branchname new_branchname` or: `git branch --move branchname new_branchname`

Show all completely merged branches with current branch: `git branch --merged`

Delete merged branch (only possible if not HEAD): `git branch -d branchname` or: `git branch --delete branchname`

Delete not merged branch: `git branch -D branch_to_delete`

## Merge

True merge (fast forward): `git merge branchname`

Merge to master (only if fast forward): `git merge --ff-only branchname`

Merge to master (force a new commit): `git merge --no-ff branchname`

Stop merge (in case of conflicts): `git merge --abort`

Stop merge (in case of conflicts): `git reset --merge` // prior to v1.7.4

Merge only one specific commit: `git cherry-pick 073791e7`

Rebase: `git checkout branchname` » `git rebase master` or: `git merge master branchname` (The rebase moves all of the commits in `master` onto the tip of `branchname`.)

Squash multiple commits into one: `git rebase -i HEAD~3` (source)

## Stash

Put in stash: `git stash save "Message"`

Show stash: `git stash list`

Show stash stats: `git stash show stash@{0}`

Show stash changes: `git stash show -p stash@{0}`

Use custom stash item and drop it: `git stash pop stash@{0}`

Use custom stash item and do not drop it: `git stash apply stash@{0}`

Delete custom stash item: `git stash drop stash@{0}`

Delete complete stash: `git stash clear`

## Gitignore & Gitkeep

About: https://help.github.com/articles/ignoring-files

Useful templates: https://github.com/github/gitignore

Add or edit gitignore: `nano .gitignore`

Track empty dir: `touch dir/.gitkeep`

## Log

Show commits: `git log`

Show oneline-summary of commits: `git log --oneline`

Show oneline-summary of commits with full SHA-1: `git log --format=oneline`

Show oneline-summary of the last three commits: `git log --oneline -3`

Show only custom commits: `git log --author="Sven"` `git log --grep="Message"` `git log --until=2013-01-01` `git log --since=2013-01-01`

Show only custom data of commit: `git log --format=short` `git log --format=full` `git log --format=fuller` `git log --format=email` `git log --format=raw`

Show changes: `git log -p`

Show every commit since special commit for custom file only: `git log 6eb715d.. index.html`

Show changes of every commit since special commit for custom file only: `git log -p 6eb715d.. index.html`

Show stats and summary of commits: `git log --stat --summary`

Show history of commits as graph: `git log --graph`

Show history of commits as graph-summary: `git log --oneline --graph --all --decorate`

## Compare

Compare modified files: `git diff`

Compare modified files and highlight changes only: `git diff --color-words index.html`

Compare modified files within the staging area: `git diff --staged`

Compare branches: `git diff master..branchname`

Compare branches like above: `git diff --color-words master..branchname^`

Compare commits: `git diff 6eb715d` `git diff 6eb715d..HEAD` `git diff 6eb715d..537a09f`

Compare commits of file: `git diff 6eb715d index.html` `git diff 6eb715d..537a09f index.html`

Compare without caring about spaces: `git diff -b 6eb715d..HEAD` or: `git diff --ignore-space-change 6eb715d..HEAD`

Compare without caring about all spaces: `git diff -w 6eb715d..HEAD` or: `git diff --ignore-all-space 6eb715d..HEAD`

Useful comparings: `git diff --stat --summary 6eb715d..HEAD`

Blame: `git blame -L10,+1 index.html`

## Releases & Version Tags

Show all released versions: `git tag`

Show all released versions with comments: `git tag -l -n1`

Create release version: `git tag v1.0.0`

Create release version with comment: `git tag -a v1.0.0 -m 'Message'`

Checkout a specific release version: `git checkout v1.0.0`

## Collaborate

Show remote: `git remote`

Show remote details: `git remote -v`

Add remote origin from GitHub project: `git remote add origin https://github.com/user/project.git`

Add remote origin from existing empty project on server: `git remote add origin ssh://root@123.123.123.123/path/to/repository/.git`

Remove origin: `git remote rm origin`

Show remote branches: `git branch -r`

Show all branches: `git branch -a`

Compare: `git diff origin/master..master`

Push (set default with `-u`): `git push -u origin master`

Push: `git push origin master`

Force-Push: `git push origin master --force`

Fetch: `git fetch origin`

Fetch a custom branch: `git fetch origin branchname:local_branchname`

Pull: `git pull`

Pull specific branch: `git pull origin branchname`

Merge fetched commits: `git merge origin/master`

Clone to localhost: `git clone https://github.com/user/project.git` or: `git clone ssh://user@domain.com/~/dir/.git`

Clone to localhost folder: `git clone https://github.com/user/project.git ~/dir/folder`

Clone specific branch to localhost: `git clone -b branchname https://github.com/user/project.git`

Delete remote branch (push nothing): `git push origin :branchname` or: `git push origin --delete branchname`

## Archive

Create a zip-archive: `git archive --format zip --output filename.zip master`

Export/write custom log to a file: `git log --author=sven --all > log.txt`

## Troubleshooting

Ignore files that have already been committed to a Git repository: http://stackoverflow.com/a/1139797/1815847

## Security

Hide Git on the web via `.htaccess`: `RedirectMatch 404 /\.git` (more info here: http://stackoverflow.com/a/17916515/1815847)

## Large File Storage

Website: https://git-lfs.github.com/

Install: `brew install git-lfs`

Track `*.psd` files: `git lfs track "*.psd"` (init, add, commit and push as written above)

**yolayne** commented on 28 Jan 2015

Small typo under "General" section:

Get **verything** ready to commit: git add .

**wreckday** commented on 11 Feb 2015

good one!

**Bhavik3** commented on 13 Feb 2015

it helps a lot for me ....thanks

**hofmannsven** commented on 16 Feb 2015 Owner

Thanks, also updated the typo 😊

**boonchu** commented on 17 Feb 2015

I thought about quick manual for git. You did it. Thanks.

**ghost** commented on 18 Mar 2015

Great work. Any of the branch stuff work for releases? When migrating for example, I am able to run (in Unix) `for remote in` git branch -r | grep -v '->' `; do git branch --track $remote; done` and then upon a push, get all of my branches to the new repo. Any shortcuts for the releases?

**SekharBeri** commented on 31 Mar 2015

Great work it helps me lot thank you

**ryrych** commented on 2 May 2015

Thanks for the info how to chop long files! It is especially useful for generated translations.js - every time you make a little change in a translation first you get hundreds of lines before and then the same amount of lines after (as there's only one line in the file)

**jfmercer** commented on 5 Jun 2015

Great work. Thanks.

**adampmoss** commented on 18 Aug 2015

Thanks for this. Bookmarked.

**tomer-ben-david** commented on 7 Sep 2015

Best git cheatsheet, bookmarked.

**tomer-ben-david** commented on 8 Sep 2015

I tend to use this alot, do you as well?

Show working tree status and ignore untracked files:
`git status -uno`

**boobiebewbs** commented on 23 Sep 2015

how do I set myself as a normal branch not a master branch?

**NZOzzDeveloper** commented on 28 Jan 2016

thanks for the cheat sheet :)

**baluragala** commented on 3 Feb 2016

Superb !!! Great Cheatsheet..

**jsroyal** commented on 6 Feb 2016

Awesome!!

**kropiv** commented on 23 May 2016

thanks a lot!

**developersamim** commented on 29 Jun 2016

very helpful... awesome tutorial

**mhdphp** commented on 3 Jul 2016

Excellent... thanks!

**kishorboddu** commented on 5 Jul 2016

thanks a lot, very helpful

**adharjain** commented on 13 Jul 2016

Awsome Help!

**tiennv90** commented on 8 Aug 2016

thank you very much for making this script

**rjangheldotcom** commented on 8 Aug 2016

nice work, thanks.

**Brentophillips** commented on 13 Oct 2016

Best list I've found, thanks! I have a large folder in a GitHub repo master branch with too many pngs inside to delete manually online. Any chance you can go through the steps, in order and for a beginner, showing how, after I've cloned my repo on my desktop, I can delete a folder and return the repo?

**Nzalo** commented on 18 Oct 2016

thanks for this 👍

**phaniapsr** commented on 11 Nov 2016

Covered all basic commands.. Thanks alot

**stevexyz** commented on 5 Dec 2016

Get everything ready to commit **including deletion and renaming**: git add **-A** .

**stevexyz** commented on 6 Dec 2016

See the last actions on the repo: "git reflog"

**khalidrizvi** commented on 26 Jan 2017

very helpful

**srinivas1787** commented on 28 Jan 2017

Awesome Sir.... Really very help ful :)

**fazlearefin** commented on 5 Feb 2017

Get everything ready to commit: `git add .`

If you are not in the base of the git repo, this command does not add all files to staging area. Better to use

```
git add -A
```

**Rongcong** commented on 9 Feb 2017

It's brilliant. Thanks for your contribution!

**jackbraj** commented on 14 Feb 2017

There is a typo in "Merge to master (**forc** a new commit): git merge --no-ff branchname".
It should be force?

**SARAN-thala** commented on 20 Feb 2017

Nice one.
It's very useful for beginners to Git.

**JaharshKotha** commented on 23 Feb 2017

What about going to different versions in the log ?

**mohammedfouad90** commented on 16 Mar 2017

Good work man .. Thanks :)

**ileontiuc** commented on 1 Apr 2017

Hi! very nice overview!
do you know if there a way of checking whether the changes from a specific commit are still maintained in another commit? (via git terminal)

**imusman** commented on 12 Apr 2017

Example Link

**sipoma** commented on 28 Apr 2017

Well Done!!! Very Very Useful

**KateGH** commented on 13 May 2017

Thank you for sharing, really useful!!

**wiseosho** commented on 30 May 2017

Thank you for sharing!.

**huoxudong125** commented on 14 Jul 2017

Git Cheat Sheets from Github

**edwinpopham** commented on 9 Aug 2017

Cool easy place to get a reminder about some commands

**parameshnalla** commented on 21 Aug 2017

Great work. Thank you

**michaeljwiebe** commented on 12 Sep 2017

Awesome, thanks for this resource! Just one question here, this seems like it can't be right since its the same command...

Commit to most recent commit: `git commit --amend -m "Message"`
Update most recent commit message: `git commit --amend -m "New Message"`

**JerryKacmar** commented on 12 Sep 2017

Great list, thanks for this.

I would add:
Show history of changes for one file: gitk [filename]

**amoljore751986** commented on 22 Sep 2017

Thanks, Very helpful

**MuraraAllan** commented on 23 Sep 2017

Thank you!

**count48** commented on 9 Oct 2017

This will be there on my bookmarks forever , thanks for putting this together **@hofmannsven**

**yh2n** commented on 18 Oct 2017

Very helpful, thanks!

**AwakenedOne** commented on 6 Nov 2017

Thanks. A huge time saver and very helpful.

**rajops** commented on 9 Nov 2017

Nice that you but together most of the Git commands, thanks 👍

**sanyog96** commented on 12 Nov 2017

Thanks

**JimmyCDChen** commented on 21 Nov 2017

Thanks!

**Arkangel17** commented on 30 Nov 2017

awesome. thanks!

**hoangthach252** commented on 5 Dec 2017

The best cheat sheets. Thanks !

**huoxudong125** commented on 20 Dec 2017

Thanks

**abdelaziz321** commented on 30 Jan

Very helpful 👍

**Valentina9091** commented on 12 Feb

Thanks a lot, very helpful

**pksjw** commented on 27 Feb

This is handy, thanks!

**cavrilionis** commented on 1 Mar

Thanks for this. Bookmarked.

**Joshhortt** commented 20 days ago

Awesome Cheatsheet, Thanks buddy!

**tech0909** commented 20 days ago

Nice!

**bestorw01** commented 20 days ago

Nice! one