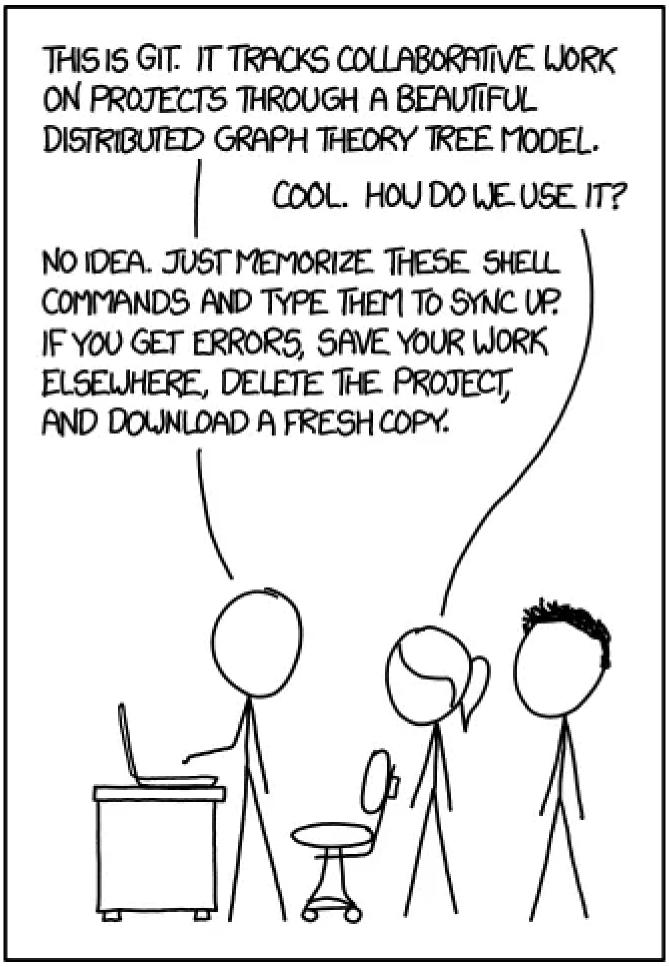
TACTIPS
Project Shrv Blog

DEVELOPER

7 Git Hacks You Just Can't Ignore

Posted on November 11, 2015 by Ritesh Shrivastav in developer

Git needs no introduction as it's the most powerful tool ever been developed to ease the life of programmers. Although proved to be a boon for open source projects, the command line interface of Git is notoriously difficult to master.


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

Usually 70% of using Git is only *add*, *commit*, *branch* and *push / pull*. Most people are familiar with the flow that always moves in one direction. Ever wondered how to go back or undo steps if you added wrong files to the *repo* or made a *commit* with wrong message to a wrong *branch*?

If you are one of those who follows what is shown in the artwork (left), then this list of hacks will make you wonder what you could have done with Git if what not and realize the possibilities.

Photo Credits: xkcd

1. How to edit an incorrect commit message?

The commit message is going to live for a very long time in your code base so, you definitely want it to be something which correctly defines the changes. This command will let you edit the most recent commit message.

You need to make sure that there are no working copy changes or they too may get committed.

```
$ git commit --amend -m "YOUR-NEW-COMMIT-MESSAGE"
```

In case you've already *pushed* your *commit* to the remote branch then you need to force push the commit with this command:

```
$ git push <remote> <branch> --force
```

You can follow this Stack Overflow answer for additional information.

2. How to undo 'git add' before commit?

What if you added some wrong files into the staging area but did not make a commit? You can undo this by a simple command. If there's only one file that needs to be removed then:

```
$ git reset <filename>
```

or if you want to unstage all due changes:

```
$ git reset
```

You can follow this Stack Overflow answer for additional information.

3. How to undo last commit?

Well its definitely not a good idea to do a second commit every now and then when you accidentally made a commit with wrong files or missed some in the first place. This three step process will have you covered in such cases.

```
$ git reset --soft HEAD~1
# make changes to your working files as necessary
$ git add -A .
$ git commit -c ORIG_HEAD
```

When you execute the first command, Git will move your HEAD pointer back to the commit you made before making this one so, now you can move files or make changes as necessary. You add all your changes and finally when you execute the last command, Git will pop your default editor with the same commit message. You may edit it if you want or you can override this by using '-C' instead of '-c' in the final command.

4. How to revert Git Repo to a previous commit?

'Reverting' can be implied in a lot of sense, but the most common is when you want a temporary revert to look around back in time (:p) and then return back to present state. This can be done by:

```
$ git checkout <SHA>
```

'*<SHA>*' is the first 8-10 characters of the Hash Code of the commit where you want to go.

It will detach the HEAD and let you fool around with no branch checked out. If you want to make commits while you're here, it can be done by creating a new branch here:

```
$ git checkout -b <SHA>
```

To go back to the present state, just checkout to the branch you were on.

You can follow this Stack Overflow answer for additional information.

5. How to undo a Git Merge?

You might have to do a *Hard Reset* to the previous commit in order to undo a merge. What 'merge' basically does is it resets the index and updates the files in the working tree that are different between *<commit>* and *HEAD*, but keeps those which are different between the index and working tree (i.e. which has changes that have not been added).

```
$ git reset --hard <SHA>
```

But there's always a second way in Git, you may explore them here.

6. How to remove local (untracked) files from current Git branch?

If you happen to have a lot of files which are untracked (because they are not required) and you don't want them to show up every time you use *git status* . There are few ways to get around this problem.

```
$ git clean -f -n
$ git clean -f
$ git clean -fd
$ git clean -fX
$ git clean -fx
```

(1): *-n* option will let you know what files will be removed if you run (2).

(2): This will remove all files as reported by command-(1).

(3): *-d* if you also want to remove directories.

(4): *-X* if you just want to remove ignored files.

(5): *-x* if you want to remove both ignored and non-ignored files

Note the case difference of *X* in last two commands.

For more information, you may explore official git-clean documentation.

7. How to delete a Git branch both locally and remotely?

To delete a local branch:

```
$ git branch --delete --force <branchName>
# OR use -D as a short-hand:
$ git branch -D
```

To delete a remote branch:

```
Shell
$ git push origin --delete <branchName>
```

Conclusion

Checkout official github training documentation for quick reference or official git docs to know all about Git.

That's it.

What are some other must-know hacks of Git?

Share this:

2

DEVELOPER GIT TACTIPS

PREVIOUS POST

Hello World! :D

LEAVE A REPLY

Your email address will not be published. Required fields are marked *

COMMENT

NAME *

EMAIL *

WEBSITE

POST COMMENT

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.