Useful git commands for everyday use!

Do you know that questions about [git](https://stackoverflow.com/tags/git) get the most views on StackOverflow? I’ve searched a lot on Google how to execute certain actions with git, and this actually slowed me down a lot. There are some actions that we tend to use a lot, so it’s good to learn them. Here are my favorites, learning from friends and internet, hope you find them useful.

Before we begin, you should run git --version to check your current git version, mine is 2.12.2 as in macOS High Sierra. [Here](https://git-scm.com/) is the official git documentation, you can read details about git commands, parameters and new releases of git.

#### Table of Contents

* [Useful commands](https://medium.com/p/e1a4de64037d#b880)
* [Git alias](https://medium.com/p/e1a4de64037d#ee7f)
* [GUI clients](https://medium.com/p/e1a4de64037d#1542)
* [Check before you commit](https://medium.com/p/e1a4de64037d#5916)
* [Where to go from her](https://medium.com/p/e1a4de64037d#1fb5)

### Useful commands

#### 🔍 Status

Check the status of working directory and staging area:

git status

Show changes between HEAD and working directory:

git diff

Show the list of commits in one line format:

git log --oneline

Show commits that make add or remove a certain string:

git log -S 'LoginViewController'

Search commits that contain a log message:

git log — all — grep=’day of week’

#### 🔍 Tag

List all tags:

git tag

Tag a commit:

git tag -a 1.4 -m "my version 1.4"

Delete remote tags:

git push --delete origin tagname

git push origin :tagname

Push tag to remote:

git push origin tagname

Rename tag:

git tag new old  
git tag -d old  
git push origin :refs/tags/old  
git push --tags

Move tag from one commit to another commit:

git push origin :refs/tags/<tagname>  
git tag -fa tagname  
git push origin master --tags

#### 🔍 Remote

List all remote:

git remote

Rename remote:

git remote rename old new

Remove stale remote tracking branches:

git remote prune origin

#### 🔍 Branch

List all branches:

git branch

Create the branch on your local machine and switch in this branch:

git checkout -b branch\_name

Create branch from commit:

git branch branch\_name sha1\_of\_commit

Push the branch to remote:

git push origin branch\_name

Rename other branch:

git branch -m old new

Rename current branch:

git branch -m new

Rename remote branch:

git branch -m old new # Rename branch locally   
git push origin :old # Delete the old branch   
git push --set-upstream origin new # Push the new branch, set local branch to track the new remote

Delete a branch:

git branch -D the\_local\_branch

git push origin :the\_remote\_branch

Delete all local branches but master

git branch | grep -v "master" | xargs git branch -D

#### 🔍 Commit

Undo last commit:

git reset --hard HEAD~1

Squash last n commits into one commit:

git rebase -i HEAD~5

git reset --soft HEAD~5  
git add .  
git commit -m "Update"  
git push -f origin master

Move last commits into new branch:

git branch newbranch  
git reset --hard HEAD~3 # Go back 3 commits. You \*will\* lose uncommitted work.\*1  
git checkout newbranch

#### 🔍 Cherry Pick

Add some commits to the top of the current branch:

git cherry-pick hash\_commit\_A hash\_commit\_B

#### 🔍 Reflog

Show reflog:

git reflog

Get commit:

git reset --hard 0254ea7

git cherry-pick 12944d8

#### 🔍 Revert

Revert the previous commit:

git revert HEAD  
git commit

Revert the changes from previous 3 commits without making commit:

git revert --no-commit HEAD~3..

#### 🔍 Amend

Amend previous commit:

git commit --amend

git commit --amend --no-edit

git commit --amend -m "New commit message"

[Changing git commit message after push](http://stackoverflow.com/questions/8981194/changing-git-commit-message-after-push-given-that-no-one-pulled-from-remote):

git commit --amend -m "New commit message"  
git push --force <repository> <branch>

#### 🔍 Checkout

Checkout a tag:

git checkout tagname

git checkout -b newbranchname tagname

Checkout a branch:

git checkout destination\_branch

Use -m if there is merge conflict:

git checkout -m master // from feature branch to master

Checkout a commit:

git checkout commit\_hash

git checkout -b newbranchname HEAD~4

git checkout -b newbranchname commit\_hash

git checkout commit\_hash file

Checkout a file:

git checkout c5f567 -- Relative/Path/To/File

#### 🔍 Stash

Save a change to stash:

git stash save "stash name"

git stash

List all stashes:

git stash list

Apply a stash:

git stash pop

git stash apply

git stash apply stash@{2}

#### 🔍 Rebase

Rebase the current branch onto master:

git rebase master // rebase the current branch onto master

Continue rebase:

git rebase --continue

Abort rebase:

git rebase --abort

#### 🔍 .gitignore

Un-track files that have just been declared in .gitignore:

git rm -r --cached .  
git add .  
git commit -am "Remove ignored files"

#### 🔍 Index

Remove untracked files:

git clean

Remove file from index:

git reset file

Reset the index to match the most recent commit:

git reset

Reset the index and the working directory to match the most recent commit:

git reset --hard

#### 🔍 Misc

Get their changes during git rebase:

git checkout --ours foo/bar.java  
git add foo/bar.java

Get their changes during git merge:

git pull -X theirs

git checkout --theirs path/to/the/conflicted\_file.php

git checkout --theirs .  
git add .

git checkout branchA  
git merge -X theirs branchB

Merge commits from master into feature branch:

git checkout feature1  
git merge --no-ff master

Find bug in commit history in a binary search tree style:

git bisect start

git bisect good

git bisect bad

### Git alias

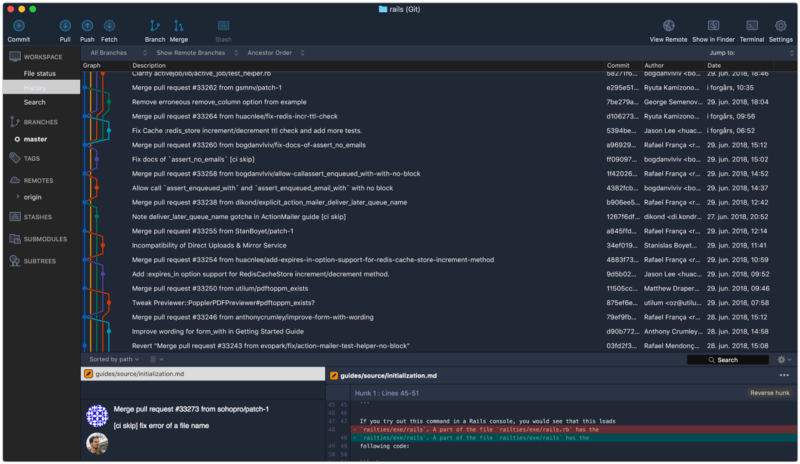
If there are commands that you use a lot, then consider using git alias. This is how to make alias for git status, then you can just type git st:

git config — global alias.st status

Alias configurations are stored in .gitconfig file, you can learn some cool aliases from [thoughtbot](https://github.com/thoughtbot/dotfiles/blob/master/gitconfig) and [mathiasbynens](https://github.com/mathiasbynens/dotfiles/blob/master/.gitconfig).

### GUI clients

Doing things in command line is cool and faster. However for viewing branches and commits, I find using a GUI client more visualizing and comfortable. You can see a list of all GUI clients [here](https://git-scm.com/download/gui/mac), I myself use [SourceTree](https://www.sourcetreeapp.com/).



### Check before you commit

We usually have some experiment code that we don’t want they to step into our commit. I usually mark my experiment with // <TEST> but sometimes forget to unstage that.

Starting with 2.9, Git has improvement on its commit hook which makes it globally using **hooksPath.**

Firstly we nee to create a file called pre-commit, and place it into, for example, /Users/khoa/hooks:

In your project, run git config core.hooksPath /Users/khoa/hooks.

Whenever you commit a file with that pattern, it won’t let you commit. For how to make this work in SourceTree, check:

[**SourceTree and pre commit hook**  
Pre-commit file works perfectly in terminal, but SourceTree seems to ignore it. I use both terminal and SourceTree, as…medium.com](https://medium.com/@onmyway133/sourcetree-and-pre-commit-hook-52545f22fe10)

### Where to go from here

This is just scratching the surface of what git can do, if you want to learn more, here are some links to get started.

* [Atlassian Git Tutorial](https://www.atlassian.com/git/tutorials/setting-up-a-repository): overview of how to set up a repository (repo) under Git version control
* [git-cheat-sheet](https://github.com/arslanbilal/git-cheat-sheet): Git cheat sheet saves you from learning all the commands by heart.
* [Learn Enough Git to Be Dangerous](http://www.learnenough.com/git-tutorial)
* [Git Workflows for Pros: A Good Git Guide](http://www.toptal.com/git/git-workflows-for-pros-a-good-git-guide)
* [Git from the inside out](https://codewords.recurse.com/issues/two/git-from-the-inside-out): The essay focuses on the graph structure that underpins Git
* [git-game](https://github.com/git-game/git-game): terminal game to test git skills
* [Introduction to Git — talk by Scott Chacon](https://www.youtube.com/watch?v=xbLVvrb2-fY)
* [Git Tutorial — Git Fu With The Command Line](http://www.raywenderlich.com/74258/git-tutorial-intermediate)
* [Git Immersion](http://gitimmersion.com/): The surest path to mastering Git is to immerse oneself in its utilities and operations, to experience it first-hand
* [git-flight-rules](https://github.com/k88hudson/git-flight-rules) Flight rules for git
* [gitflow](https://github.com/nvie/gitflow) Git extensions to provide high-level repository operations for Vincent Driessen’s branching model
* [diff-so-fancy](https://github.com/so-fancy/diff-so-fancy) Good-lookin’ diffs with diff-highlight and more
* [github-cheat-sheet](https://github.com/tiimgreen/github-cheat-sheet) A list of cool features of Git and GitHub
* [git tips](https://github.com/git-tips/tips) Most commonly used git tips and tricks
* [Little Things I Like to Do with Git](https://csswizardry.com/2017/05/little-things-i-like-to-do-with-git/)