How do I alias commands in git?

Asked 9 years, 6 months ago Active 3 months ago Viewed 209k times

582

I saw a screencast where someone had gotten

```
git st
git ci
```

to work. When I do it I get an error asking me if I meant something else.
Being a git newb, I need to know what you have to do to get this done?

178

git

asked Mar 31 '10 at 14:31

DevelopingChris
21.4k 26 81 116

1 You can also see it here git-scm.com/book/en/v2/Git-Basics-Git-Aliases – JayRizzo Nov 21 '17 at 17:43

Also see further questions on more advanced usage of git alias here: stackoverflow.com/questions/46528736/… – NeilG Aug 22 at 4:01

20 Answers

917

Basically you just need to add lines to `~/.gitconfig`

```
[alias]
    st = status
    ci = commit -v
```

Or you can use the git config alias command:

```
$ git config --global alias.st status
```

On unix, use single quotes if the alias has a space:

```
$ git config --global alias.ci 'commit -v'
```

On windows, use double quotes if the alias has a space or a command line argument:

```
c:\dev> git config --global alias.ci "commit -v"
```

The alias command even accepts functions as parameters. Take a look at aliases.

edited May 19 '16 at 14:01

Warren P
41.7k 34 155 286

answered Mar 31 '10 at 14:33

Diego Dias
16.7k 5 28 34

82 I highly recommend you use `git config --global` to place the aliases in `~/.gitconfig` instead of `.git/config` for your current repository. – Cascabel Mar 31 '10 at 14:56

25 I prefer settings `st` to `status -s` (short status) – hasen Mar 31 '10 at 15:59

18 This is really awesome. I have been looking for this. Just a heads up, if you have a command with spaces you should use `'` like `git config --global alias.sr 'svn rebase'` – Amir Raminfar Dec 1 '11 at 19:39

1 @HellishHeat These aliases are created by git, for git. If you want aliases for some other command line system, you'll have to look up how to do that one that system. (You appear to be using a Unix-like system, and I happen to know that creating aliases on Unices is quite simple. The syntax is different though. Try a Google search.) – Michael Dorst Oct 13 '14 at 16:01

12 Just another heads up, if you're using Git on Windows command line, then you will need to use double quotes `"` instead of single quotes when adding command with spaces, e.g. `git config --global alias.ci "commit -v"` – ABVincita Aug 6 '15 at 5:01

167

As others have said the appropriate way to add git aliases is in your global `.gitconfig` file either by editing `~/.gitconfig` or by using the `git config --global alias.<alias> <git-command>` command

Below is a copy of the alias section of my `~/.gitconfig` file:

```
[alias]
    st = status
    ci = commit
```

```
    co = checkout
    br = branch
    unstage = reset HEAD --
    last = log -1 HEAD
```

Also, if you're using bash, I would recommend setting up bash completion by copying `git-completion.bash` to your home directory and sourcing it from your `~/.bashrc`. (I believe I learned about this from the Pro Git online book.) On Mac OS X, I accomplished this with the following commands:

```
# Copy git-completion.bash to home directory
cp usr/local/git/contrib/completion/git-completion.bash ~/

# Add the following lines to ~/.bashrc
if [ -x /usr/local/git/bin/git ]; then
    source ~/.git-completion.bash
fi
```

Note: The bash completion will work not only for the standard git commands but also for your git aliases.

Finally, to really cut down on the keystrokes, I added the following to my `~/.bash_aliases` file, which is sourced from `~/.bashrc`:

```
alias gst='git status'
alias gl='git pull'
alias gp='git push'
alias gd='git diff | mate'
alias gau='git add --update'
alias gc='git commit -v'
alias gca='git commit -v -a'
alias gb='git branch'
alias gba='git branch -a'
alias gco='git checkout'
alias gcob='git checkout -b'
alias gcot='git checkout -t'
alias gcotb='git checkout --track -b'
alias glog='git log'
alias glogp='git log --pretty=format:"%h %s" --graph'
```

edited Mar 31 '10 at 15:26

answered Mar 31 '10 at 15:19

Matthew Rankin

296k 35 108 148

2 For linux, I did this to get the git-completion.bash stuff: blogs.oracle.com/linuxnstuff/entry/… – Duncan Lock Mar 9 '12 at 15:25

9 If you use zsh, the excellent oh-my-zsh suite contains a plugin with all those "standard" git aliases - github.com/robbyrussell/oh-my-zsh/blob/master/plugins/git/… -- for bash, have a look at github.com/revans/bash-it – jobwat Aug 26 '13 at 0:22

noob question: what does it mean to "be sourced from" `~/.bashrc` file? – ahnbizcad Aug 25 '15 at 6:27

1 @ahnbizcad: See tldp.org/HOWTO/Bash-Prompt-HOWTO/x237.html – Matthew Rankin Aug 25 '15 at 15:38

1 `~/.bashrc` : to really cut down the key-strokes. Exactly what was looking for. – parasrish Sep 20 '17 at 8:39

62

I think the most useful gitconfig is like this,we always use the 20% function in git,you can try the "g ll",it is amazing,the details:

```
[user]
    name = my name
    email = me@example.com
[core]
    editor = vi
[alias]
    aa = add --all
    bv = branch -vv
    ba = branch -ra
    bd = branch -d
    ca = commit --amend
    cb = checkout -b
    cm = commit -a --amend -C HEAD
    ci = commit -a -v
    co = checkout
    di = diff
    ll = log --pretty=format:"%C(yellow)%h%Cred%d\\ %Creset%s%Cblue\\ [%cn]" --decorate
--numstat
    ld = log --pretty=format:"%C(yellow)%h\\ %C(green)%ad%Cred%d\\ %Creset%s%Cblue\\
[%cn]" --decorate --date=short --graph
    ls = log --pretty=format:"%C(green)%h\\ %C(yellow)[%ad]%Cred%d\\ %Creset%s%Cblue\\
[%cn]" --decorate --date=relative
    mm = merge --no-ff
    st = status --short --branch
    tg = tag -a
    pu = push --tags
    un = reset --hard HEAD
    uh = reset --hard HEAD^
   [color]
    diff = auto
    status = auto
    branch = auto
```

```
[branch]
 autosetuprebase = always
```

edited Feb 28 '17 at 14:03

Orangetronic
190 3 10

answered Feb 20 '14 at 14:12

wcc526
2,125 2 23 27

how do you set this up? what do you put where to make this so? – ahnbizcad Aug 25 '15 at 5:57

3 @ahnbizcad Place in ~/.gitconfig if you git config --global otherwise it goes in .git/config of current repository – jared Apr 27 '16 at 12:55

16

You need the `git config alias` command. Execute the following in a Git repository:

```
git config alias.ci commit
```

For global alias:

```
git config --global alias.ci commit
```

answered Mar 31 '10 at 14:34

Alan Haggai Alavi
61.5k 14 90 121

9

This worked for me:

```
bco = "!f(){ git branch ${1} && git checkout ${1}; };f"
```

on:

```
$ git --version

git version 1.7.7.5 (Apple Git-26)
```

answered Sep 24 '12 at 18:08

Nicolas Gramlich
2,704 15 17

1 you could also do: git config --global alias.bco 'checkout -b'. Then you could do: git bco new-branch. :) – Russell Feb 12 '13 at 22:53

3 I like `git cob` . reminds me of summer, as in corn on the cob. actually a great word we don't think about enough... cob that is – Мати Тернер Feb 26 '14 at 18:22

4 In case this is the first time anyone other than me has seen a git alias command starting with `!` , note that `Since version 1.5.0, Git supports aliases executing non-git commands, by prefixing the value with "!"` (ref) – Sam Aug 15 '14 at 11:13

7

This will create an alias `st` for `status` :

```
git config --add alias.st status
```

answered Mar 31 '10 at 14:34

Nick Moore
12.2k 5 52 79

I needed the --add and to use double quotes, not single quotes – Aligned Dec 18 '15 at 17:06

Why `git st` when you can use `git s` , get rid of that `s` :P – Abdullah Apr 12 '17 at 11:19

6

You can alias both git and non-git commands. It looks like this was added in version 1.5. A snippet from the `git config --help` page on version 2.5.4 on my Mac shows:

> If the alias expansion is prefixed with an exclamation point, it will be treated as a shell command.

For example, in your global `.gitconfig` file you could have:

```
[alias]
    st = status
    hi = !echo 'hello'
```

And then run them:

```
$ git hi
hello
$ git st
On branch master

...
```

answered Feb 4 '16 at 17:38

mkobit
25.3k 6 103 117

6

Follwing are the 4 git shortcuts or aliases youc an use to save time.

Open the commandline and type these below 4 commands and use the shortcuts after.

```
git config --global alias.co checkout
git config --global alias.ci commit
git config --global alias.st status
git config --global alias.br branch
```

Now test them!

```
$ git co              # use git co instead of git checkout
$ git ci              # use git ci instead of git commit
$ git st              # use git st instead of git status
$ git br              # use git br instead of git branch
```

answered Mar 3 '15 at 5:23

Prabhakar Undurthi
4,687 33 39

```
$ git update
git: 'update' is not a git command. See 'git --help'.
```

4

Did you mean this?

```
    update-ref

$ git config --global alias.update 'pull -v'

$ git update
From git://git.kernel.org/pub/scm/git/git
 = [up to date]      html       -> origin/html
 = [up to date]      maint      -> origin/maint
 = [up to date]      man        -> origin/man
 = [up to date]      master     -> origin/master
 = [up to date]      next       -> origin/next
 = [up to date]      pu         -> origin/pu
 = [up to date]      todo       -> origin/todo
Already up-to-date.
```

answered Mar 31 '10 at 14:35

Greg Bacon
107k 27 172 228

For those looking to **execute shell commands in a git alias**, for example:

4

```
$ git pof
```

In my terminal will *push force* the current branch to my origin repo:

```
[alias]
    pof = !git push origin -f $(git branch | grep \\* | cut -d ' ' -f2)
```

Where the

```
$(git branch | grep \\* | cut -d ' ' -f2)
```

command returns the current branch.

So this is a shortcut for manually typing the branch name:

```
git push origin -f <current-branch>
```

answered Apr 7 '17 at 20:00

Gus
2,127 1 17 25

Why not "simply" `git push -f origin HEAD` to push current branch to its remote counterpart? Also, a shortcut to push with force? If you have to push force frequently enough to benefit from a shortcut, isn't something amiss elsewhere in your setup or workflow? – RomainValeri Jan 17 at 17:13

Bash meshed up creating the alias (replacing `!git` with the last git command), but manually editing the config file did the trick. – H. de Jonge Sep 27 at 9:09

3

Add the following lines to your ~/.gitconfig in your home directory

```
[alias]
# one-line log
l = log --pretty=format:"%C(yellow)%h\\ %ad%Cred%d\\ %Creset%s%Cblue\\ [%cn]" --decorate
--date=short
ll = log --pretty=format:"%C(yellow)%h%Cred%d\\ %Creset%s%Cblue\\ [%cn]" --decorate --
numstat
ld = log --pretty=format:"%C(yellow)%h\\ %C(green)%ad%Cred%d\\ %Creset%s%Cblue\\ [%cn]"
--decorate --date=short --graph
ls = log --pretty=format:"%C(green)%h\\ %C(yellow)[%ad]%Cred%d\\ %Creset%s%Cblue\\
[%cn]" --decorate --date=relative

a = add
ap = add -p
c = commit --verbose
ca = commit -a --verbose
cm = commit -m
cam = commit -a -m
m = commit --amend --verbose

d = diff
ds = diff --stat
dc = diff --cached

s = status -s
co = checkout
cob = checkout -b
# list branches sorted by last modified
b = "!git for-each-ref --sort='-authordate' --format='%(authordate)%09%
(objectname:short)%09%(refname)' refs/heads | sed -e 's-refs/heads/--'"
```

```
# list aliases
la = "!git config -l | grep alias | cut -c 7-"
```

Once that is done, you can do `git a` instead of `git add` for example. The same applies to other commands under the alias heading..

answered Oct 19 '18 at 16:02

Stryker
2,402 24 44

2

I created alias `dog` for showing the log graph:

```
git config --global alias.dog "log --oneline --graph --all --decorate"
```

And use it as follow:

```
git dog
```

answered Jul 18 at 10:04

Seyed Morteza Mousavi
3,283 6 33 56

2

You can set custom git aliases using git's config. Here's the syntax:

```
git config --global alias.<aliasName> "<git command>"
```

For example, if you need an alias to display a list of files which have merge conflicts, run:

```
git config --global alias.conflicts "diff --name-only --diff-filter=U"
```

Now you can use the above command only using "conflicts":

```
git conflicts
# same as running: git diff --name-only --diff-filter=U
```

edited Apr 5 '18 at 11:57
ACHERONFAIL
1,801 2 15 39

answered Apr 5 '18 at 11:26
Akshay Bosamiya
39 6

2

Just to get the aliases even shorter than the standard git config way mentioned in other answers, I created an npm package mingit (`npm install -g mingit`) so that most commands would become 2 characters instead of 2 words. Here's the examples:

```
g a .                   // git add .
g b other-branch        // git branch other-branch
g c "made some changes" // git commit -m "made some changes"
g co master             // git checkout master
g d                     // git diff
g f                     // git fetch
g i                     // git init
g m hotfix              // git merge hotfix
g pll                   // git pull
g psh                   // git push
g s                     // git status
```

and other commands would be similarly short. This also keeps bash completions. The package adds a bash function to your dotfiles, works on osx, linux, and windows. Also, unlike the other aliases, it aliases `git` -> `g` as well as the second parameter.

edited Aug 25 '16 at 21:58

answered Nov 17 '15 at 2:24
ironicaldiction
717 3 9 21

1 Thank you for creating the github project. – biniam May 24 '16 at 15:47

1

For me (**I'm using mac with terminal**) only worked when I added on **.bash_profile** and opened another tab to load the change:

```
alias gst="git status"
alias gd="git diff"
alias gl="git log"
alias gco="git commit"
alias gck="git checkout"
alias gl="git pull"
alias gpom="git pull origin master"
```

```
alias gp="git push"
alias gb="git branch"
```

answered Mar 13 at 14:39

Marina
668 9 17

1

[PFA screenshot of my .gitconfig file]

with the below aliases

```
[alias]
    cb = checkout branch
    pullb = pull main branch
```

answered Nov 23 '18 at 8:48

SRK
26 6

1

You can also chain commands if you use the '!' operator to spawn a shell:

```
aa = !git add -A && git status
```

This will both add all files and give you a status report with `$ git aa` .

For a handy way to check your aliases, add this alias:

```
alias = config --get-regexp ^alias\\.
```

Then a quick `$ git alias` gives you your current aliases and what they do.

answered Jan 30 '17 at 21:38

Joseph Cheek
349 3 8

1

It is given here Aliases.Even there are great answers here, I added this because it differs in windows and linux

answered Aug 15 '13 at 3:14

madhu131313
3,268 5 26 45

0

Another possibility for windows would be to have a directory filled with .bat files that have your shortcuts in them. The name of the file is the shortcut to be used. Simply add the directory to your PATH environment variable and you have all the shortcuts to your disposal in the cmd window.

For example (gc.bat):

```
git commit -m %1
```

Then you can execute the following command in the console:

```
gc "changed stuff"
```

The reason I'm adding this as an answer is because when using this you aren't limited to `git ...` only commands.

answered Mar 23 '17 at 23:50

Rick van Osta
856 6 18

0

If you want an alternative to the `~/.gitconfig` option and open to digging in a little more, another option is to write entirely custom git commands by wrapping them in a global node package.

In your package.json, you'd define the root command (example: `gt`), and then filter the specific commands to execute the correct git commands. For example, `git checkout my-branch` could be `gt co mybranch` .

The "christian-git" package on npm uses this method: https://github.com/alexmacarthur/christian-git

answered Sep 12 '17 at 12:30

Alex MacArthur
1,232 9 14