

## commands

:Git [args]	Similar to :!git [args] but chdir to the repository tree first.
:Git! [args]	Like :Git , but capture the output into a temp file
:Gcd [dir]	:cd relative to the repository.
:Glcd [dir]	:lcd relative to the repository.
:Gstatus	Bring up the output of git-status in the preview window.
:Gcommit [args]	If there is nothing to commit, :Gstatus is called instead.
:Gmerge [args]	Calls git-merge and loads errors and conflicted files
:Gpull [args]	Like :Gmerge , but for git-pull.
:Gpush [args]	Invoke git-push, load the results into the quickfix
:Gfetch [args]	Like :Gpush , but for git-fetch.
:Ggrep [args]	:grep with git-grep as 'grepprg'.
:Glgrep [args]	:lgrep with git-grep as 'grepprg'.
:Glog [args]	Load all previous revisions of the current file into the quickfix list.
:{range}Glog [args]	Use git-log -L to load previous revisions of the given range of the current file into the quickfix list.
:Gllg [args]	Like :Glog , but use the location list instead of the quickfix list.
:Gedit [revision]	:edit a fugitive-revision .
:Gsplit [revision]	:split a fugitive-revision .
:Gvsplit [revision]	:vsplit a fugitive-revision .
:Gtabedit [revision]	:tabedit a fugitive-revision .
:Gpedit [revision]	:pedit a fugitive-revision .

:Gsplit! [args]	Like :Git! , but open the resulting temp file in a
:Gvsplit! [args]	^ split, tab, or preview window.
:Gtabedit! [args]	^
:Gpedit! [args]	^
:Gread [revision]	Empty the buffer and :read a fugitive-revision.
:{range}Gread [revision]	:read in a fugitive-revision after {range}.
:Gread! [args]	Empty the buffer and :read the output of a Git command.
:{range}Gread! [args]	:read the output of a Git command after {range}.
:Gwrite	Write to the current file's path and stage the results.
:Gwrite {path}	You can give :Gwrite an explicit path of where in the work tree to write. You can also give a path like :0:foo.txt or even :0 to write to just that stage in the index.
:Gwq [path]	Like :Gwrite followed by :quit if the write succeeded.
:Gwq! [path]	Like :Gwrite! followed by :quit! if the write succeeded.
:Gdiff [revision]	
:Gsdiff [revision]	Like :Gdiff, but always split horizontally.
:Gvdiff [revision]	Like :Gdiff, but always split vertically.
:Gmove {destination}	Add a ! to pass -f
:Gremove	Add a ! to pass -f
:Gblame [flags]	Run git-blame on the file
:{range}Gblame [flags]	Run git-blame on the given range.
:{range}Gbrowse	
:{range}Gbrowse!	put the URL on the clipboard rather than opening it.
:{range}Gbrowse {revision}	Like :Gbrowse, but for a given fugitive-revision. A useful value here is -, which ties

## SPECIFYING REVISIONS

master	.git/refs/heads/master
HEAD	.git/HEAD
HEAD^{}	The commit referenced by HEAD
HEAD^	The parent of the commit referenced by HEAD
HEAD:	The tree referenced by HEAD
/HEAD	The file named HEAD in the work tree
Makefile	The file named Makefile in the work tree
HEAD^:Makefile	The file named Makefile in the parent of HEAD
:Makefile	The file named Makefile in the index (writable)
-	The current file in HEAD
^	The current file in the previous commit
~3	The current file 3 commits ago
:	.git/index (Same as :Gstatus)
:0	The current file in the index
:1	The current file's common ancestor during a conflict
:2	The current file in the target branch during a conflict
:3	The current file in the merged branch during a conflict
:/foo	The most recent commit with "foo" in the message

## mappings

<C-R><C-G>	On the command line, recall the path to the current object
["x]y<C-G>	Yank the commit SHA and path to the current object.

### available in Git objects.

<CR>	Jump to the revision under the cursor.
o	Jump to the revision under the cursor in a new split.
S	Jump to the revision under the cursor in a new vertical split.
O	Jump to the revision under the cursor in a new tab.
-	Go to the tree containing the current tree or blob.
~	Go to the current file in the [count]th first ancestor.
P	Go to the current file in the [count]th parent.
C	Go to the commit containing the current file.
.	Start a command line with the current revision prepopulated at the end of the line.
a	Show the current tag, commit, or tree in an alternate format.

:*[range]*Gbrowse  
[...]*@{remote}*

the URL to the latest commit rather than a volatile branch.

Force using the given remote rather than the remote for the current branch. The remote is used to determine which GitHub repository to link to.

## Gstatus commands

g?	show this help
<C-N>	next file
<C-P>	previous file
<CR>	:Gedit
-	:Git add
-	:Git reset (staged files)
cA	:Gcommit --amend --reuse-message=HEAD
ca	:Gcommit --amend
cc	:Gcommit
cva	:Gcommit --amend --verbose
cvc	:Gcommit --verbose
D	:Gdiff
ds	:Gsdiff
dp	:Git! diff (p for patch; use :Gw to apply)
dp	:Git add --intent-to-add (untracked files)
dv	:Gvdiff
O	:Gtabedit
o	:Gsplit
p	:Git add --patch
p	:Git reset --patch (staged files)
q	close status
r	reload status
S	:Gvsplit

## Gblame

g? | show help

### open

o | open commit in horizontal split  
O | open commit in new tab

### close

q | close blame and return to blamed window  
gq | q, then :Gedit to return to work tree version  
<CR> | q, then open commit

### resize

A | resize to end of author column  
C | resize to end of commit column  
D | resize to end of date/time column

### reblame

- | reblame at commit  
~ | reblame at [count]th first grandparent  
P | reblame at [count]th parent (like HEAD^[count])