


git add, commit and push commands in one?

 stackoverflow.com/questions/19595067/git-add-commit-and-push-commands-in-one

Is there any way to use these three commands in one?

```
git add .
git commit -a -m "commit" (do not need commit message either)
git push
```

Sometimes I'm changing only one letter, CSS padding or something. Still, I have to write all three commands to push the changes. There are many projects where I'm only one pusher, so this command would be awesome!

git

18 Answers

up vote 140
down vote

Building off of @Gavin's answer:

Making lazygit a function instead of an alias allows you to pass it an argument. I have added the following to my .bashrc (or .bash_profile if Mac):

```
function lazygit() {
    git add .
    git commit -a -m "$1"
    git push
}
```

This allows you to provide a commit message, such as

```
lazygit "My commit msg"
```

You could of course beef this up even more by accepting even more arguments, such as which remote place to push to, or which branch.

answered Apr 27 '14 at
21:06



btse
5,23121823

[up vote](#) 44 [down vote](#)

I ended up adding an alias to my `.gitconfig` file:

```
[alias]
  cmp = "!f() { git add -A && git commit -m \"\$@\" && git push; }; f"
```

Usage: `git cmp "Long commit message goes here"`

Adds all files, then uses the comment for the commit message and pushes it up to origin.

I think it's a better solution because you have control over what the commit message is.

The alias can be also defined from command line, this adds it to your `.gitconfig` :

```
git config --global alias.cmp '!f() { git add -A && git commit -m \"\$@\" && git push; }; f'
```

[up vote](#) 33 [down vote](#)

While I agree with Wayne Werner on his doubts, this is technically an option:

```
git config alias.acp '! git commit -a -m "commit" && git push'
```

Which defines an alias that runs `commit` and `push`. Use it as `git acp`. Please be aware that such "shell" aliases are always run from the root of your git repository.

Another option might be to write a post-commit hook that does the push.

answered Oct 25 '13 at 17:06



[Tilman Vogel](#)
4,90932425

[up vote](#) 27 [down vote](#)

I think you might misunderstand the workflow that git was designed for. (To clarify/correct what I meant in the comment, you don't need the `git add .`, since `commit -a` usually serves the same purpose - adding any changes that have not yet been staged, if the files have already been added)

Typically you'll do something like this:

```
# make some changes
$ git commit -a -m "Changed something"
# make some more changes
$ git commit -a -m "Changed something else"
```

wash, rinse, repeat, until you've finished feature X, or you're at a stopping point, or you just want other people to see what you've done. Then you do

```
$ git push
```

Git is not SVN - but it appears that you're trying to use it as such. You might find some of the resources at the end of the article [here](#) to be of some use.

up vote 15 down vote I use this on my .bash_profile

```
gitpush() {  
    git add .  
    git commit -m "$*"   
    git push  
}  
alias gp=gitpush
```

It executes like

gp A really long commit message

Don't forget to run `source ~/.bash_profile` after saving the alias.

answered Jan 2 '16 at
19:39



Rajender Joshi
2,819,115,311

up vote 14 down vote

Set as an alias in bash:

```
$ alias lazygit="git add .; git commit -a -m '...'; git push;";
```

Call it:

```
$ lazygit
```

(To make this alias permanent, you'd have to include it in your .bashrc or .bash_profile)

up vote 14 down vote

You can use bash script , set alias to launch any command or group of commands

```
git commit -am "your message" && git push
```

up vote
6
down vote

You can try gitu.

For the first time (node js has to be installed):

```
npm install -g git-upload
```

After that:

```
gitu COMMIT_MSG
```

To issue those three commands at once.

The good thing is that you **don't have to worry when you reinstall your system or when you want to do this on different computers and No file modification is needed.** This also **work on different platforms** (not just Linux and Mac, but also Windows under command prompt like `cmd` and `powershell`) just that you have to install `npm` and `nodejs` (`git` of course).

up vote 5
down vote

If the file is already being tracked then you do not need to run `git add`, you can simply write `git commit -am 'your message'`

If you do not want to write a commit message you might consider doing something like

```
git commit --allow-empty-message -am ''
```

answered Oct 26 '13 at
23:48



Peter Foti
4,60332237

up vote 3
down vote

As mentioned in [this](#) answer, you can create a **git alias** and assign a set of commands for it. In this case, it would be:

```
git config --global alias.add-com-push '!git add . && git commit -a -m "commit" && git push'
```

and use it with

```
git add-com-push
```

up vote 2 down vote

I use a batch file:

```
@ECHO OFF
SET /p comment=Comment:
git add *
git commit -a -m
"%comment%"
git push
```

answered Mar 31 '16 at
15:16



Márcio Souza Júnior
416414

up
vote 1
down
vote

There are some issues with the scripts above:

shift "removes" the parameter \$1, otherwise, "push" will read it and "misunderstand it".

My tip :

```
git config --global alias.acpp '!git add -A && branchatu="$(git symbolic-ref HEAD 2>/dev/null)" && branchatu=${branchatu##refs/heads/} && git commit -m "$1" && shift && git pull -u origin $branchatu && git push -u origin $branchatu'
```

up vote 1 down
vote

When you want svn-like behavior of git commit, use this in your git aliases in your .gitconfig

```
commit = "!f() { git commit \"\$@\" && git push; };f"
```

answered Aug 16 '16 at
11:30



Rik van der Heijden
112

up vote 1 down
vote

If you want to edit your commit message better, I mean add more than one lines etc... then

```
git commit -a && git push
```

will open the editor and once you save the message it will also push it.

answered Dec 18 '17 at
21:35



mCeviker
3681620

up vote 0 down vote

I did this .sh script for command

```
#!/bin/sh
cd LOCALDIRECTORYNAME/
git config --global user.email
"YOURMAILADDRESS"
git config --global user.name
"YOURUSERNAME"
git init
git status
git add -A && git commit -m
"MESSAGEFORCOMMIT"
git push origin master
```

answered Mar 31 '17 at
12:32



acs
305626

up
vote
0
down
vote

Since the question doesn't specify which shell, here's the eshell version based on the earlier answers. This goes in the eshell alias file, which might be in ~/.emacs.d/eshell/alias I've added the first part z <https://github.com/rupa/z/> which let's you quickly cd to a directory, so that this can be run no matter what your current directory is.

```
alias census z cens; git add .; git commit -m "fast"; git push
```

answered Aug 3 '17 at 9:47



בנימין הגלילי
3991717

up vote
0
down
vote

Add in ~/.bash_profile for adding, committing and pushing with one command put:

```
function g() { git commit -a -m "$*"; git push; }
```

Usage:

```
g your commit message  
g your commit message 'message'
```

No quotes are needed although you can't use semicolons or parenthesis in your commit messages (single quotes are allowed). If you want to any of these just simply put double quotes in you message, e.g.:

```
g "your commit message; (message)"
```

To create a comment in your message do:

```
g "your commit message:  
> your note"
```

There's also a function for adding and committing in a similar way:

```
function c() { git add --all; git commit -m "$*"; }
```

Works exactly the same way that **g** function and has the same constraints. Just put **c** instead. E.g.

```
c your commit message
```

You can also add an alias for pushing to the remote:

```
alias p='git push'
```

Usage:

```
p
```

That amounts into 2 letters, **c** and **p** you use while working with your git repository. Or you can use **g** instead to do it all with only one letter.

Full list of aliases and functions:

<https://gist.github.com/matt360/0c5765d6f0579a5aa74641bc47ae50ac>

answered Feb 5 at
16:07



Carrot Cake
1

[up](#)
[vote](#) 0
[down](#)
[vote](#)

Building off the [lazygit](#) answer, the following solution adds a user check to verify the changes before pushing. It will revert the commands if cancelled. And all that will happen if and only if there are changes in the local repo.

```
### SAFER LAZY GIT
function lazygit() {
  git add .
  if git commit -a -m "$1"; then
    read -r -p "Are you sure you want to push these changes? [y/N]" response
    case "$response" in
      [yY][eE][sS]|[yY])
        git push
        ;;
      *)
        git reset HEAD~1 --soft
        echo "Reverted changes."
        ;;
    esac
  fi
}
```




answered Feb 10 at
23:46



[Matthew Cordaro](#)
331220

Your Answer

Sign up or log in

-  Sign up using Google
-  Sign up using Facebook
-  Sign up using Email and Password

Post as a guest

By posting your answer, you agree to the [privacy policy](#) and [terms of service](#).

Not the answer you're looking for? Browse other questions tagged [git](#) or ask your own question.
