**8 Git aliases that make me more efficient**

By Ricardo Gerardi (Red Hat)6-8 minutes

The excellent article [*7 Git tricks that changed my life*](https://opensource.com/article/20/10/advanced-git-tips) inspired me to write about another Git feature that's had a major impact on my experience using Git on the command line: aliases.

Defining Git aliases to serve as substitutes for commands provides two major benefits:

* It simplifies long commands that have many options, making them shorter and easier to remember.
* It shortens frequently used commands so that you can work more efficiently.

**How to define and use aliases**

To define a Git alias, use the git config command with the alias and the command you want to substitute. For example, to create the alias p for git push:

$ git config --global alias.p 'push'

You can use an alias by providing it as an argument to git, just like any other command:

To see all your aliases, list your configuration with git config:

$ git config --global -l

user.name=ricardo

user.email=ricardo@example.com

alias.p=push

You can also define aliases with your favorite shell, such as Bash or Zsh. However, defining aliases using Git offers several features that you don't get using the shell. First, it allows you to use aliases across different shells with no additional configuration. It also integrates with Git's autocorrect feature, so Git can suggest aliases as alternatives when you mistype a command. Finally, Git saves your aliases in the user configuration file, allowing you to transfer them to other machines by copying a single file.

Regardless of the method you use, defining aliases improves your overall experience with Git. For more information about defining Git aliases, take a look at the [Git Book](https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases).

**8 useful Git aliases**

Now that you know how to create and use an alias, take a look at some useful ones.

**1. Git status**

Git command line users often use the status command to see changed or untracked files. By default, this command provides verbose output with many lines, which you may not want or need. You can use a single alias to address both of these components: Define the alias st to shorten the command with the option -sb to output a less verbose status with branch information:

$ git config --global alias.st 'status -sb'

If you use this alias on a clean branch, your output looks like this:

Using it on a branch with changed and untracked files produces this output:

$ git st

## master

 M test2

?? test3

**2. Git log --oneline**

Create an alias to display your commits as single lines for more compact output:

$ git config --global alias.ll 'log --oneline'

Using this alias provides a short list of all commits:

$ git ll

33559c5 (HEAD -> master) Another commit

17646c1 test1

**3. Git last commit**

This shows details about the most recent commit you made. This extends an example in the Git Book's chapter on [Aliases](https://git-scm.com/book/en/v2/Git-Basics-Git-Aliases):

$ git config --global alias.last 'log -1 HEAD --stat'

Use it to see the last commit:

$ git last

commit f3dddcbaabb928f84f45131ea5be88dcf0692783 (HEAD -> branch1)

Author: ricardo <ricardo@example.com>

Date:   Tue Nov 3 00:19:52 2020 +0000

    Commit to branch1

 test2 | 1 +

 test3 | 0

 2 files changed, 1 insertion(+)

**4. Git commit**

You use git commit a lot when you're making changes to a Git repository. Make the git commit -m command more efficient with the cm alias:

$ git config --global alias.cm 'commit -m'

Because Git aliases expand commands, you can provide additional parameters during their execution:

$ git cm "A nice commit message"

[branch1 0baa729] A nice commit message

 1 file changed, 2 insertions(+)

**5. Git remote**

The git remote -v command lists all configured remote repositories. Shorten it with the alias rv:

$ git config --global alias.rv 'remote -v'

**6. Git diff**

The git diff command displays differences between files in different commits or between a commit and the working tree. Simplify it with the d alias:

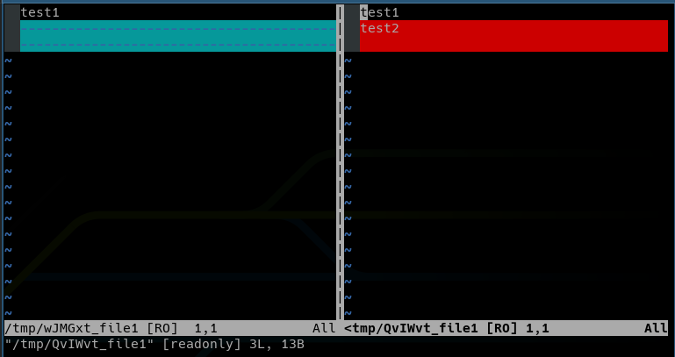
$ git config --global alias.d 'diff'

The standard git diff command works fine for small changes. But for more complex ones, an external tool such as vimdiff makes it more useful. Create the alias dv to display diffs using vimdiff and use the -y parameter to skip the confirmation prompt:

$ git config --global alias.dv 'difftool -t vimdiff -y'

Use this alias to display file1 differences between two commits:

$ git dv 33559c5 ca1494d file1



**7. Git config list**

The gl alias makes it easier to list all user configurations:

$ git config --global alias.gl 'config --global -l'

Now you can see all defined aliases (and other configuration options):

$ git gl

user.name=ricardo

user.email=ricardo@example.com

alias.p=push

alias.st=status -sb

alias.ll=log --oneline

alias.last=log -1 HEAD --stat

alias.cm=commit -m

alias.rv=remote -v

alias.d=diff

alias.dv=difftool -t vimdiff -y

alias.gl=config --global -l

alias.se=!git rev-list --all | xargs git grep -F

**8. Git search commit**

Git alias allows you to define more complex aliases, such as executing external shell commands, by prefixing them with the ! character. You can use this to execute custom scripts or more complex commands, including shell pipes.

For example, define the se alias to search within your commits:

$ git config --global alias.se '!git rev-list --all | xargs git grep -F'

Use this alias to search for specific strings in your commits:

$ git se test2

0baa729c1d683201d0500b0e2f9c408df8f9a366:file1:test2

ca1494dd06633f08519ec43b57e25c30b1c78b32:file1:test2

**Autocorrect your aliases**

A cool benefit of using Git aliases is its native integration with the autocorrect feature. If you make a mistake, by default Git suggests commands that are similar to what you typed, including aliases. For example, if you type ts instead of st for status, Git will suggest the correct alias:

$ git ts

git: 'ts' is not a git command. See 'git --help'.

The most similar command is

        st

If you have autocorrect enabled, Git will automatically execute the correct command:

$ git config --global help.autocorrect 20

$ git ts

WARNING: You called a Git command named 'ts', which does not exist.

Continuing in 2.0 seconds, assuming that you meant 'st'.

## branch1

?? test4

**Optimize Git commands**

Git alias is a useful feature that improves your efficiency by optimizing the execution of common and repetitive commands. Git allows you to define as many aliases as you want, and some users define many. I prefer to define aliases for just my most used commands—defining too many makes it harder to memorize them and may require me to look them up to use them.

For more about aliases, including other useful ones, see the [Git Wiki's Aliases page](https://git.wiki.kernel.org/index.php/Aliases).

[Creative Commons License](http://creativecommons.org/licenses/by-sa/4.0/)This work is licensed under a Creative Commons Attribution-Share Alike 4.0 International License.