GIT: From Beginner To Fearless

GIT Aliasing Activity:
Using Aliases to simplify commands

Brian Gorman, Author/Instructor/Trainer

Introduction

Git has a lot of flexibility and a lot of commands. However, some commands are pretty verbose, while others are hard to remember, and still others are just tedious to type over and over again. For this reason GIT provides a way for us to setup aliases to use in our day-to-day operations.

One thing to remember about using aliases is that if you change your machine you will have to recreate your aliases. Additionally, unless you purpose to do so, you likely will never have a tracked file containing your aliases. For this reason, if you rely on a lot of aliases and want to make sure that you don't ever lose them, I would recommend creating a private repository and simply tracking your global config file alias settings in that repository. Then, if you change machines or something else goes wrong, your aliases will not be lost forever.

In this activity, we'll take a quick look at creation and use of a few aliases in GIT.

Let's gets started!

Step 1: Working with aliases

Before creating any new aliases, it would be a great idea to make sure they aren't already there. If they are and you added it again, it would just overwrite anyway, but it's nice to know if you already have them or be able to list them at any time.

a) Determine current aliases.

[**git config --global --list**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global --list
user.email=blgorman@gmail.com
user.name=Brian L. Gorman
core.autocrlf=true
core.editor='C:\Program Files (x86)\Microsoft VS Code\code.exe' -w
core.excludesfile=C:/Users/Brian/.gitIgnore
credential.helper=manager
diff.tool=code
difftool.code.cmd=code --wait --diff $LOCAL $REMOTE
difftool.prompt=false
merge.tool=code
mergetool.code.cmd=code --wait $MERGED
mergetool.prompt=false
mergetool.keepbackup=false
alias.onelinegraph=log --oneline --graph --decorate
alias.expireunreachablenow=reflog expire --expire-unreachable=now --all
alias.gcunreachablenow=gc --prune=now
fetch.prune=true
push.followtags=true
```

I have three aliases set at this time. If I want to see just my aliases, I can do one of two commands:

[**git config --global --list | grep alias**]

or

[**git config --get-regexp alias**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global --list | grep alias
alias.onelinegraph=log --oneline --graph --decorate
alias.expireunreachablenow=reflog expire --expire-unreachable=now --all
alias.gcunreachablenow=gc --prune=now

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global --get-regexp alias
alias.onelinegraph log --oneline --graph --decorate
alias.expireunreachablenow reflog expire --expire-unreachable=now --all
alias.gcunreachablenow gc --prune=now
```

b) Add an alias.

Before we add any aliases, remember that if you set aliases in the global config they are going to be machine specific. For this reason, you may wish to backup your global config somewhere, as using an alias you rely upon on another machine won't be possible until it's added. This could be especially problematic for you if you create an alias for a complex command, use your alias for a couple of years, then lose your alias. By that time, you may have forgotten what the underlying command was and it may be nearly un-recoverable for you.

Notes

The syntax to add an alias is fairly straight-forward [also remember that if the alias is already there it will get overwritten with the new value].

Add an entry to the global config for
`alias.<your-alias-name>`
followed by the command without 'git' in it and wrapped in quotes.

For example, adding an alias for git checkout master called 'chkmstr' would look like this:

[**git config --global alias.chkmstr 'checkout master'**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global alias.chkmstr 'checkout master'
```

[**git config --global --get alias.chkmstr**] //to see it

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global --get alias.chkmstr
checkout master
```

Then just run the command with the keyword git followed by your alias:

[**git chkmstr**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git chkmstr
Already on 'master'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git checkout -b anotherBranch
Switched to a new branch 'anotherBranch'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (anotherBranch)
$ git chkmstr
Switched to branch 'master'
```

c) Add any alias(es) you would like

Here I want to challenge you to pick a couple of commands you would like to alias. For this exercise, keep it to a single command. I'm going to show you how to set up a multiple command alias in just a bit.
//here are my three aliases listed as above:

[**git config --global alias.onelinegraph**
'log --oneline --graph --decorate']

[**git config --global --get alias.onelinegraph**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global alias.onelinegraph 'log --oneline --graph --decorate'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global --get alias.onelinegraph
log --oneline --graph --decorate
```

[**git config --global alias.expireunreachablenow**
'reflog expire --expire-unreachable=now --all']
[**git config --global --get alias.expireunreachablenow**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global alias.expireunreachablenow 'reflog expire --expire-
unreachable=now --all'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global --get alias.expireunreachablenow
reflog expire --expire-unreachable=now --all
```

[**git config --global alias.gcunreachablenow 'gc --prune=now'**]
[**git config --global --get alias.gcunreachablenow**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global alias.gcunreachablenow 'gc --prune=now'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DefaultWeb (master)
$ git config --global --get alias.gcunreachablenow
gc --prune=now
```

Step 2: Working with aliases that take variables

As you might already be thinking, perhaps I want to set a variable and use an alias with a variable, such as 'git checkout -b <some-new-branch>. Git has made this possible.

a) Add an alias that allows checking out a variable branch name

The syntax for the alias creation is exactly the same. The only difference is that now we need to add a $n variable where n is the number of the parameter. To add an alias that lets you checkout a new branch at any time, we need to invoke the shell, and we MUST pass the 'git' keyword along with the command now:

[**git config --global alias.chknewbr**
'!sh -c "git checkout -b $1"']
[**git config --global --get alias.chknewbr**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$ git config --global alias.chknewbr '!sh -c "git checkout -b $1"'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$ git config --global --get alias.chk
alias.chkmstr    alias.chknewbr

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$ git config --global --get alias.chknewbr
!sh -c "git checkout -b $1"
```

now invoke it:

[**git chknewbr 'brian-testing'**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (master)
$ git chknewbr 'brian-testing'
Switched to a new branch 'brian-testing'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (brian-testing)
$
```

Step 3: Aliases that take variables and multiple commands.

To finish up our activity, let's see what happens if we need to run multiple commands and also take a variable

a) Create an alias for checking out master, fetching origin, pulling master to local, then switching to a variable branch and merging master!

[**git config --global alias.setuplocalbr '!sh -c "git checkout master && git fetch origin && git pull origin master && git checkout $1 && git merge origin master**]

[**git config --global –get alias.setuploclbr**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (brian-testing)
$ git config --global alias.setuplocalbr '!sh -c "git checkout master && git fet
ch origin && git pull origin master && git checkout $1 && git merge master"'

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (brian-testing)
$ git config --global --get alias.setuplocalbr
!sh -c "git checkout master && git fetch origin && git pull origin master && git
 checkout $1 && git merge master"
```

Run it:

[**git steuplocalbr brian-testing**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (brian-testing)
$ git setuplocalbr brian-testing
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.
From https://github.com/majorguidancesolutions/SimpleActivityRepo
 * branch            master     -> FETCH_HEAD
Already up-to-date.
Switched to branch 'brian-testing'
Already up-to-date.

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/GitRevertDemo (brian-testing)
$
```

This concludes our look at working with aliases in GIT

Closing Thoughts

Git aliasing is powerful and effective. By setting up a few simple aliases we can easily make our day-to-day command line operations easier to manage.

In this activity, we set up a few simple aliases, and then also saw how we can create aliases that run multiple commands as well as take positional parameters for execution.

Take a few minutes to make some notes about the various commands we've learned about in this activity, and practice using them.

Notes