GIT: From Beginner to Fearless

GIT Default Difftool Activity: Setting our difftool to VSCode

Brian Gorman, Author/Instructor/Trainer

Introduction

Although BASH allows us to see differences in the terminal, working in the command line can be a bit tedious. This is especially true when trying to discern the differences in files across commits in the command line. To make viewing differences more attractive, a very nice option is to use Visual Studio Code as the difftool.

Once VSCode is installed and set as the difftool, we'll be able to easily compare the differences between a couple of commits, or the difference between a commit and our working branch.

This activity assumes you have previously completed the activity "Setting VSCode as our default editor" If you have not done that activity, you should complete it before starting this activity.

Let's gets started!

Step 1: Go to any repository

a) Make sure you are on any working repository. It doesn't even have to be up to date. We are not going to be affecting anything.

Our goal is to get our global config to have three entries for difftool:

```
diff.tool=code
difftool.code.cmd=code --wait --diff $LOCAL $REMOTE
difftool.prompt=false
```

Additionally, after completion of this activity, if we were to look at our file with our editor, we will see entries like this:

```
[diff]
    tool = code
[difftool "code"]
    cmd = code --wait --diff $LOCAL $REMOTE
[difftool]
    prompt = false
```

On the working repository

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

//I've removed it for now to do this activity:

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (ma
$ 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
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 --a
alias.gcunreachablenow=gc --prune=now
alias.chkmstr=checkout master
alias.chknewbr=!sh -c "git checkout -b $1"
alias.setuplocalbr=!sh -c "git checkout master && git fetch origin &&
rigin master && git checkout $1 && git merge master"
```

b) Set and verify the difftool variable in our global config

[**git config --global diff.tool code**]

[**git config --global --get diff.tool**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git config --global diff.tool code

Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git config --global --get diff.tool
code
```

Notes

c) Set the actual execution command for code as the difftool, and verify:

[**git config --global difftool.code.cmd 'code --wait --diff $LOCAL $REMOTE'**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git config --global difftool.code.cmd 'code --wait --diff $LOCAL $REMOTE'
```

[**git config --global --get difftool.code.cmd**]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFolder/RebasingActivity1 (master)
$ git config --global --get difftool.code.cmd
code --wait --diff $LOCAL $REMOTE
```

d) Optional, turn off the annoying prompt for every diff to open:

[**git config –global difftool.prompt false**]

```
blgor@CORSAIRONE MINGW64 /d/Data/Test_Run/Defaultv
aster)
$ git config --global difftool.prompt false
```

Step 3: Review the actual file entries

a) Open the file and find the entries

[git config --global –e]

```
Brian@SENTINEL MINGW64 /g/Data/GFBTF/DemoFold
$ git config --global -e
```

```
[diff]
    tool = code
[difftool "code"]
    cmd = code --wait --diff $LOCAL $REMOTE
[difftool]
    prompt = false
```

b) Optional – Move the entries in the file.

I don't want these on the bottom. I'm going to move them to be between credential manager and merge tool {don't worry if you don't have a mergetool set yet…that's another activity…}

```
[credential]
    helper = manager
[diff]
    tool = code
[difftool "code"]
    cmd = code --wait --diff $LOCAL $REMOTE
[difftool]
    prompt = false
[merge]
    tool = code
```

The order makes absolutely no difference, it is up to you what you want it to be. Just remember that this order determines what you see on --list from global config.

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

```
credential.helper=manager
diff.tool=code
difftool.code.cmd=code --wait --diff $LOCAL $REMOTE
difftool.prompt=false
merge.tool=code
```

Step 4: Make sure the difftool works!

a) Need to be on a repository, change any file

Just open any tracked file and make changes:

[**git status**]

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   info.txt
```

[**git difftool head**]

It's working!

b) Show modified and staged changes in difftool

[**git add .**]

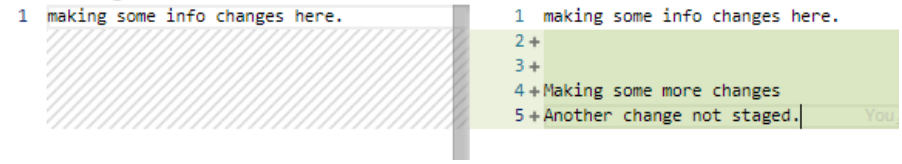
[**code info.txt**] or [**vim info.txt**]

[**git status**]

```
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage

        modified:   info.txt

Changes not staged for commit:
  (use "git add <file>..." to update what wi
  (use "git checkout -- <file>..." to discar

        modified:   info.txt
```

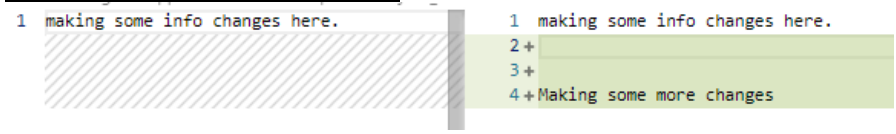
[**git difftool HEAD**]

```
Brian@SENTINEL MINGW64 /g
$ git difftool head
```



[**git difftool --cached**]

```
Brian@SENTINEL MINGW64 /g/D
$ git difftool --cached
```



c) Here we can either just commit the changes or reset back to where we were.

If you are on an important repo that you have current changes you want, don't run this command, instead, just check in your changes or undo the unwanted changes manually. If you are in a playground, then run the following

[**git reset --hard head**]

```
aster)
$ git reset --hard HEAD
HEAD is now at 60d7558 added an info.txt file
```

[we'll cover resetting later in the course, but this just reverted the repository to the state which it was in at the last commit]

This concludes our GIT difftool Activity.

Closing Thoughts

Setting VSCode [or another tool] to use for the difftool is very powerful and effective. If you don't mind seeing the changes in the console, you don't need to do this. Also note even with the difftool you can still use the console by just using the default 'diff' command rather than the difftool command we setup in this activity.

Also, as an FYI, once you have this setup, it will override default settings for your machine's difftool. If you are using Visual Studio and want to continue to use that tool for your difftool, then you will just want to unset these global variables to revert back to Visual Studio's default diff tool.

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

Notes