Configuring diff tool with .gitconfig

Asked 13 years, 6 months ago Modified 4 months ago Viewed 322k times

228

How do I configure Git to use a different tool for diffing with the `.gitconfig` file?

I have this in my .gitconfig:

```
[diff]
    tool = git-chdiff #also tried /bin/git-chdiff
```

It does not work; it just opens the regular command line diff. When I do:

```
export GIT_EXTERNAL_DIFF=git-chdiff
```

then `git diff` will open up the external diffing tool (so I know the external diff tool script works fine). Do I have something wrong with my .gitconfig configuration for the diff tool?

git

Share
Improve this question
Follow

edited Aug 16 at 17:18
Samuel RIGAUD
1,700 ● 1 ● 17 ● 25

asked Jun 20, 2011 at 14:08
ryanzec
28k ● 39 ● 115 ● 173

1 See stackoverflow.com/questions/255202/… (possibly close this as duplicate)
– Stein G. Strindhaug Jun 20, 2011 at 14:16

2 Related posts for variety of diff tools available in market - Use BeyondCompare to see difference between files in GIT, Git: How configure KDiff3 as merge tool and diff tool, use Winmerge inside of Git to file diff, Setting up and using Meld as your git difftool and mergetool, Configuring a diff-tool for Git on Windows – RBT Apr 5, 2018 at 23:44

11 Answers

Sorted by: Highest score (default)

240

An additional way to do that (from the command line):

```
git config --global diff.tool tkdiff
git config --global merge.tool tkdiff
git config --global --add difftool.prompt false
```

The first two lines will set the difftool and mergetool to `tkdiff` - change that according to your preferences. The third line disables the annoying prompt so whenever you hit `git difftool` it will automatically launch the difftool.

Share

Improve this answer

Follow

edited Jul 20, 2018 at 23:27
Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Nov 21, 2013 at 14:06
Omer Dagan
15.9k ● 16 ● 47 ● 61

14 I did `git config --global diff.tool diffmerge` but when i did `git diff myfile.txt` it still gave me the default unix diff – amphibient Jan 31, 2014 at 19:24

Perhaps you have a local configuration that defines another diff tool? – Omer Dagan Feb 2, 2014 at 8:10

1 perhaps you need to pass correct parameters to diffmerge: this link explains how to do it for meld: link : [nathanhoad.net/how-to-meld-for-git-diffs-in-ubuntu-hardy]() – Puttaraju Apr 15, 2014 at 5:41

10 @amphibient: try git diff**tool** myfile.txt to use the tool. – JBRWilkinson Apr 6, 2016 at 15:09

@amphibient <or rather anyone reading this 10 years later> see the [readme]() of a project I got recommended in a question of mine. TL; DR: the option is called `core.pager` but is not as simple as a program name. – Vorac Oct 9 at 1:55

173

Git offers a range of difftools pre-configured "out-of-the-box" (kdiff3, kompare, tkdiff, meld, xxdiff, emerge, vimdiff, gvimdiff, ecmerge, diffuse, opendiff, p4merge and araxis), and also allows you to specify your own. To use one of the pre-configured difftools (for example, "vimdiff"), you add the following lines to your `~/.gitconfig`:

```
[diff]
    tool = vimdiff
```

Now, you will be able to run `git difftool` and use your tool of choice.

Specifying your own difftool, on the other hand, takes a little bit more work, see [How do I view 'git diff' output with my preferred diff tool/ viewer?]()

Share

Improve this answer

Follow

edited Aug 17 at 19:26
Samuel RIGAUD
1,700 ● 1 ● 17 ● 25

answered Jun 20, 2011 at 14:18
Fredrik Pihl
45.6k ● 7 ● 88 ● 133

1 What do you mean by a pre-configured "out-of-the-box" difftool? To set-up an external diff tool "winMerge" which is not in your list I had to do the very same setting which you have mentioned in your post and everything started to work without any additional configuration. So does that mean that git supports "winMerge" as well out of the box because as much as I can

understand from your post that it requires some extra work/settings/configuration to setup a diff tool which is not supported out-of-the box by git. – RBT Mar 23, 2017 at 2:30

98

Others have done a 99% answer on this, but there is one step left out. (My answer will be coming from OS X so you will have to change file paths accordingly.)

You make these changes to your `~/.gitconfig`:

```
[diff]
    tool = diffmerge
[difftool "diffmerge"]
    cmd = /Applications/Diffmerge.app/Contents/MacOS/diffmerge $LOCAL $REMOTE
```

This will fix the diff tool. You can also fix this without editing the `~/.gitconfig` directly by entering these commands from the terminal:

```
git config --global diff.tool diffmerge
git config --global difftool.diffmerge.cmd
"/Applications/DiffMerge.appContents/MacOS/diffmerge \$LOCAL \$REMOTE"
```

The 1% that everyone else failed to mention is when using this you can't just run `git diff myfile.txt`; you need to run **`git difftool myfile.txt`**.

Share

Improve this answer

Follow

edited Aug 21 at 13:38
Samuel RIGAUD
1,700 ● 1 ● 17 ● 25

answered Feb 4, 2015 at 18:57
tgoza
1,111 ● 8 ● 7

27 Upvoted for that last 1%. There is nothing as satisfactory as a fully hammered nail ;) – Titou Apr 13, 2016 at 11:54

1 Is there any way to make it the default tool? – math0ne Jul 9, 2017 at 8:26

10 Great, You are missing the step `git config --global --add difftool.prompt false`, though. ;) – Suma Feb 28, 2018 at 11:48

45

Reproducing my answer from this question which was more specific to setting Beyond Compare as diff tool for Git. All the details that I've shared are equally useful for any diff tool in general, so I am sharing it here.

The first command that we run is as below:

```
git config --global diff.tool bc3
```

The above command creates below entry in **.gitconfig** file found in **%userprofile%** directory:

```
[diff]
    tool = bc3
```

%userprofile% is an environment variable which you can type on **Run** prompt and hit Enter to open the directory location where **.gitconfig** file is present.

That's it. This is all you required while setting up an already published version of any well-known comparison tool which is already known to Git like in this case 3rd version of Beyond Compare is known to Git.

Let's do a deep-dive now!

Additionally, you *might* be required to run the below command also:

```
git config --global difftool.bc3.path "c:/program files/beyond compare
3/bcomp.exe"
```

Running this command is *optional*. It is required in some specialized cases only. We will know its reason in a short while from now.

The *most important* thing to know here is the key **bc3**. This is a well-known key to Git which maps to a particular version of a specific comparison tool available in market e.g. in this case **bc3** corresponds to 3rd version of Beyond Compare tool. If you want to see complete list of the keys maintained by Git then run below comand on Git Bash command-line:

```
git difftool --tool-help
```

When we run above command then it returns below list:

```
vimdiff
vimdiff2
vimdiff3
araxis
bc
bc3
codecompare
deltawalker
diffmerge
diffuse
ecmerge
emerge
examdiff
gvimdiff
gvimdiff2
```

```
gvimdiff3
kdiff3
kompare
meld
opendiff
p4merge
tkdiff
winmerge
xxdiff
```

While setting up a comparison tool for Git, we can use any of the above pre-existing keys based on the tool and its version you're using e.g. for Beyond Compare v1 we'll use the key **bc**, for Beyond Compare v3 we'll use the key **bc3**.

But in some cases, we might have to define a brand new key of our own e.g. let's say we're setting-up a brand new comparison tool which has just been released to market. For obvious reasons the current version of Git installed on your machine will not show any key corresponding to this new tool. Eventually Git will show it in a future release but not immediately. Similarly, this problem can occur for a newly released version of an existing tool also e.g. there is no key for Beyond Compare v4 in above list. So you are always free to map any tool to any of pre-existing keys or to a *new* custom key of your own.

Now let us understand below scenarios for while setting up a comparison tool which is:

- **A new version of an old tool has got released which is not mapped to any pre-defined keys in Git**?

OR

- **Absolutely new in market**

Like in my case, I had installed Beyond Compare v4. Beyond Compare tool is already known to Git but its version 4 release is not mapped to any of the existing keys. So we can follow any of the below approaches:

1. Since no key exists in Git for Beyond Compare v4 so we can map it to the already existing key **bc3** even though it is meant to be mapped to Beyond Compare v3. We can outsmart Git to save some effort.

   Now here is the answer to the question we left unanswered in the first paragraph - If you map any tool to the key which is already known to Git then you would *not* need to run the second command. This is because the tool's EXE location is already known to Git.

   But remember, this will work only when the EXE location of the tool doesn't change across versions. If Beyond Compare v3 and v4 have different install locations in %programfiles% directory then it becomes mandatory to run the second command.

For e.g. if I had installed Beyond Compare v3 on my box then having below configuration in my **.gitconfig** file would have been sufficient to complete the setup process. This is based on the assumption that v3 and v4 will have same install path.

```
[diff]
tool = bc3
```

But if we want to associate the non-default tool then we need to mention the **path** attribute separately so that Git will know the path of EXE from where it has to be launched. Below entry tells Git to launch Beyond Compare v4 instead. Note the EXE's path:

```
[difftool "bc3"]
path = c:/program files/Beyond Compare 4/bcomp.exe
```

Also, if we wanted we could have mapped Beyond Compare v4 to *any* of the pre-defined keys of other tools as well e.g. **examdiff**. Git won't stop you from doing this bad thing. Although we should *not* do so to avoid a maintenance nightmare.

2. **Cleanest approach is to define a custom key**. We can define a brand new key for any new comparison tool or a new version of an old tool. Like in my case I defined a new key **bc4** as it is fairly intuitive. I could have named it **foobaar**.

   Now when the key is absolutely new, the setup process is slightly different. In this case you have to run two commands in all. But our second command will not be setting path of our new tool's EXE. Instead we have to set **cmd** attribute for our new tool as shown below:

```
git config --global diff.tool bc4

git config --global difftool.bc4.cmd "\"C:\\Program Files\\Beyond
Compare 4\\bcomp.exe\" -s \"\$LOCAL\" -d \"\$REMOTE\""
```

   Running above commands creates below entries in your **.gitconfig** file:

```
[diff]
tool = bc4

[difftool "bc4"]
cmd = \"C:\\Program Files\\Beyond Compare 4\\bcomp.exe\" -s \"$LOCAL\"
-d \"$REMOTE\"
```

I would strongly recommend you to follow approach # 2 to avoid any maintenance issues in future.

Share

Improve this answer

Follow

edited May 14 at 5:31

answered Aug 31, 2017 at 6:19

RBT

25.8k ● 22 ● 175 ● 255

37

Here's the part of my ~/.gitconfig where I configure diff and merge tools. I like diffmerge by SourceGear. (I like it very very much, as a matter of fact).

```
[merge]
        tool = diffmerge
[mergetool "diffmerge"]
        cmd = "diffmerge --merge --result=\"$MERGED\" \"$LOCAL\" \"$(if
test -f \"$BASE\"; then echo \"$BASE\"; else echo \"$LOCAL\"; fi)\"
\"$REMOTE\""
        trustExitCode = false
[diff]
        tool = diffmerge
[difftool "diffmerge"]
        cmd = diffmerge \"$LOCAL\" \"$REMOTE\"
```

So, you see, you're defining a tool named "diffmerge" in the `[difftool "diffmerge"]` line. Then I'm setting the tool "diffmerge" as the default in the `[diff]  tool =` section.

I obviously have the "diffmerge" command in my path, here. Otherwise I'd need to give a full path to the executable.

Share Improve this answer Follow

answered Jun 20, 2011 at 14:24

Dan Ray

21.9k ● 7 ● 64 ● 88

Without all the backslash escaping: `diffmerge --merge --result="$MERGED" "$LOCAL" "$(if test -f "$BASE"; then echo "$BASE"; else echo "$LOCAL"; fi)" "$REMOTE"` – Tim Lovell-Smith Feb 13, 2015 at 22:43

1 I'm going with the simpler `diffmerge --merge --result="$MERGED" "$LOCAL" "$BASE" "$REMOTE"` – Tim Lovell-Smith Feb 13, 2015 at 22:51

And it's got a handy 'sgdm_cygwin.sh' script to make the Windows version easy to use from Cygwin. – thoni56 Feb 27, 2016 at 10:29

So you can just do `ln-s <pathtodiffmerge>/sgdm_cygwin.sh /usr/local/bin/diffmerge` and you're all set to use the config above. As well as `diffmerge` directly from the cygwin command line with cygwin paths. – thoni56 Feb 27, 2016 at 10:39

It would be `sgdm.exe` nowadays – kzu Aug 26, 2019 at 20:50

19

Adding one of the blocks below works for me to use KDiff3 for my Windows and Linux development environments. It makes for a nice consistent cross-platform diff and merge tool.

Linux

```
[difftool "kdiff3"]
    path = /usr/bin/kdiff3
    trustExitCode = false
[difftool]
    prompt = false
[diff]
    tool = kdiff3
[mergetool "kdiff3"]
    path = /usr/bin/kdiff3
    trustExitCode = false
[mergetool]
    keepBackup = false
[merge]
    tool = kdiff3
```

Windows

```
[difftool "kdiff3"]
    path = C:/Progra~1/KDiff3/kdiff3.exe
    trustExitCode = false
[difftool]
    prompt = false
[diff]
    tool = kdiff3
[mergetool "kdiff3"]
    path = C:/Progra~1/KDiff3/kdiff3.exe
    trustExitCode = false
[mergetool]
    keepBackup = false
[merge]
    tool = kdiff3
```

Share
Improve this answer
Follow

edited Jul 20, 2018 at 23:29
Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Jan 23, 2015 at 2:43
moodboom
6,873 ● 3 ● 46 ● 47

Have you found a way to make a single cross platform .gitconfig file that specifies the right paths conditionally for both linux and windows? – Jerry Asher Sep 30, 2018 at 7:59

1 Honestly, I've switched to meld everywhere, and it just seems to work. – moodboom Oct 1, 2018 at 7:38

Is there a reason to set trustExitCode = false ? From my understanding if false is set, git would ignore non zero exit code from your tool. So if your tool crush git would mistakenly think

the merge is successful. But may be there is a reason for setting it false, please explain.
– apollo Jan 21, 2019 at 23:53

I believe `trustExitCode = false` will cause git to ask you if the merge was successful, rather than relying on whether the tool happened to exit with a true or false status.
– moodboom Jan 22, 2019 at 0:22

6

If you want to have an option to use multiple diff tools, add an alias to file *.gitconfig*:

```
[alias]
    kdiff = difftool --tool kdiff3
```

Share
Improve this answer
Follow

edited Aug 30, 2021 at 18:46
Peter Mortensen
31.6k ● 22 ● 109 ● 133

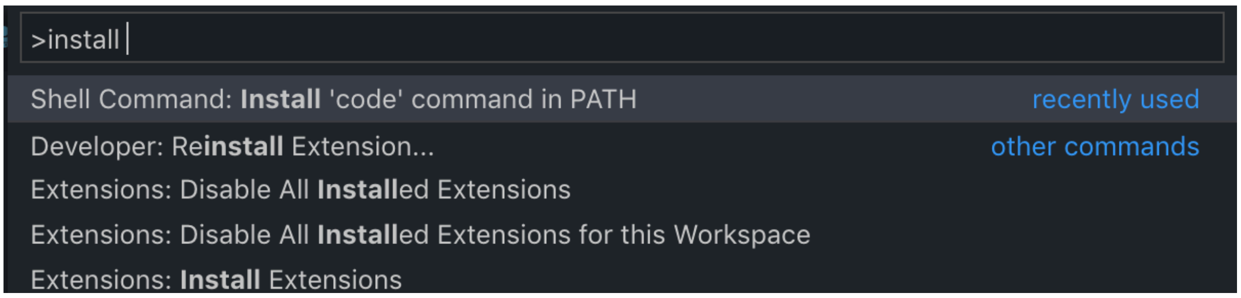
answered Oct 19, 2018 at 1:33
Alexander Katz
119 ● 1 ● 4

4

Refer to Microsoft's *VS Code Tips and Tricks*. Just run these commands in your terminal:

`git config --global merge.tool code`

But firstly you need add the `code` command to your PATH environment variable.



```
>install
Shell Command: Install 'code' command in PATH                recently used
Developer: Reinstall Extension...                            other commands
Extensions: Disable All Installed Extensions
Extensions: Disable All Installed Extensions for this Workspace
Extensions: Install Extensions
```

Share
Improve this answer
Follow

edited Aug 30, 2021 at 18:44
Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Mar 1, 2019 at 17:26
sudoz
3,543 ● 1 ● 23 ● 19

The code command is already on my user path, but this did not work for me. – Matthew Oct 15, 2021 at 16:39

2

In Windows we need to run the `git difftool --tool-help` command to see the various options like:

```
    'git difftool --tool=<tool>' may be set to one of the following:
                    vimdiff
                    vimdiff2
                    vimdiff3

    The following tools are valid, but not currently available:
                    araxis
                    bc
                    bc3
                    codecompare
                    deltawalker
                    diffmerge
                    diffuse
                    ecmerge
                    emerge
                    examdiff
                    gvimdiff
                    gvimdiff2
                    gvimdiff3
                    kdiff3
                    kompare
                    meld
                    opendiff
                    p4merge
                    tkdiff
                    winmerge
                    xxdiff

Some of the tools listed above only work in a windowed
environment. If run in a terminal-only session, they will fail.
```

And we can add any of them (for example, WinMerge) like

```
git difftool --tool=winmerge
```

For configuring Notepad++ to see files before committing:

```
 git config --global core.editor "'C:/Program
Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"
```

And using `git commit` will open the commit information in Notepad++.

Share
Improve this answer
Follow

edited Aug 30, 2021 at 18:36
Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Jul 21, 2019 at 10:43
Goyal Vicky
1,289 ● 16 ● 16

Almost all the solutions in the previous answers doesn't work with Git version 2

1

Mine: Git version = 2.28.0

Solution of the difftool: **git config --global diff.tool vimdiff**

After it, you can use it without any problems.

Share
Improve this answer
Follow

edited Aug 30, 2021 at 18:41
Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Jan 20, 2021 at 17:31
ILIASS B.
463 ● 5 ● 9

0

Use `git config --global diff.external git-chdiff` instead of `git config --global diff.tool git-chdiff`.

Share Improve this answer Follow

answered Apr 4, 2023 at 12:55
Pelican Cyber
43 ● 1 ● 7

Your answer could be improved with additional supporting information. Please edit to add further details, such as citations or documentation, so that others can confirm that your answer is correct. You can find more information on how to write good answers in the help center. – Community Bot Apr 8, 2023 at 11:19