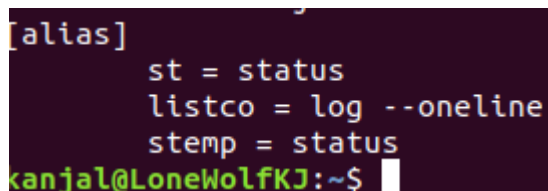


first undo git alias. there would usually be no reason to do so because as mentioned in the call even basically all the standard git commands including the one you just set an alias for would still work.

there are a couple of ways to do it, the following is what i do and then will mention a easy way too

- 1) if while setting the alias you used the --global flag, (eg - git config --global --alias.st status) then what you can do is in your home directory (this term would be discussed tomorrow if you are not familiar with it, i know i used it a couple of times during the call too, can ignore for now. basically the folder/directory you go to when you type the command cd ~) you would find a file named '.gitconfig' (a hidden file). you can open that via your text editor and just remove the line mentioning your alias (it would be clearly visible once you open the file) the lines would look like the following.



```
[alias]
    st = status
    listco = log --oneline
    stemp = status
kanjal@LoneWolfKJ:~$
```

- 2) If you did not use the --global flag, then go to the 'config' file present in the .git folder of your git repository and over there you would find similar lines as above, remove them.
- 3) The Easy way
git config --global --unset alias.st (basically alias name after the '.', also remove the --global flag if you did not use it while setting it. This is why i don't like this way, you usually wouldn't remember if you had set it globally or locally and would end up running the same command twice once with the global flag and next time without it, reason being if you set the alias for that repository only, and you use the --global flag while removing it in this way it would not work and also no error would be shown (so in many cases you might assume that it worked and continue on)) [well depends on your choice i guess. Both ways work.]

.gitignore

There is no concept, it just tells git which files/folders to ignore for staging (when you do the git add command you are adding the files/folders for staging right, so files/folders that are present in the .gitignore or match the pattern present in .gitignore they would be excluded)

An alternative would be to manually not add those files when using git add command, just add files you want to commit and push. But usually while working on a project with many files/folders internally. It would be tedious to do manual adding and so what is generally done is git add --all or git add . (git add . basically means add all folders and files including sub-folders/files present inside the current folder/directory where i am present, more would be discussed on this '.' or '..' etc tomorrow, '.' basically means current directory where you

are). So now if we are going to use the `git add --all` or `git add .` then even the files we want excluded would be added for staging right, so `.gitignore` does that for us, it prevents those files from being added to staging. This is also why files that are already staged or committed or pushed once would not be ignored. (it stops from adding to staging, once you have already added it, then it won't prevent it, there are ways to un-stage, or revert commits or remove that files. Those are details i think you can figure out later if you face them or ping any of the mentors for more info)