Sessions 11-15

#11 - Learn Everything About Aliases

- Git aliases are **a powerful workflow tool that create shortcuts to frequently used Git commands**. Using Git aliases will make you a faster and more efficient developer
- For example lets say that we want to cut short the ‘git status’ command with just ‘st’

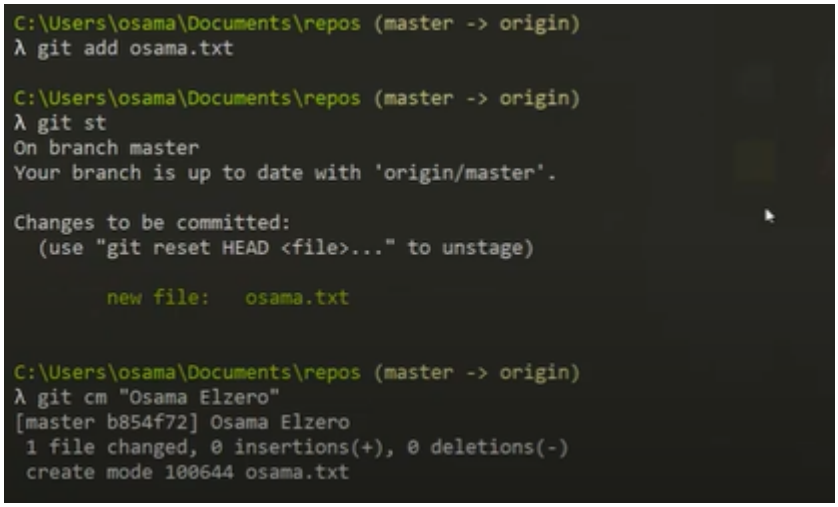
```
git status
```

- To do that, we make the aliases on the global configuration not just on our work environment.

```
git config --global aliases.st status
#To make sure that alias took 'st' and assigned for 'status'
$ git config --global alias.st
status #result
```

- To make a shortcut for a command that has spaces in it. Lets take as an example the ‘commit’ command.

```
git config --global alias.cm "commit -m"
```



- We can get predefined aliases by people did before, that would ease our work.

#12 - Branching And Merging

- Git branch is a development section that is created when you want to make add or modify code on your project without affecting directly the main branch.
- If you work in a company you make create a branch to enhance part of the project’s code or add on it. Then you make a pull request so that it can be merged with the main branch
- To create a branch named ‘Scroll’ for example

```
git branch Scroll
```

- To check your branches

```
git branch
```

- To move from branch ‘main’ to ‘scroll’

```
git checkout Scroll
```

- To make a safe delete which allows git to check if there is any modification on your branch before deletion. Consequently, if there is any modifications git will not delete your branch
- For safe delete use flag ‘-d’. To force delete use flag ‘-D’

```
git branch -d Scroll
```

- Lets say we want to move in a branch called ‘Dev-Feature’ and create it at the same time

```
git checkout -b Dev-Feature
```

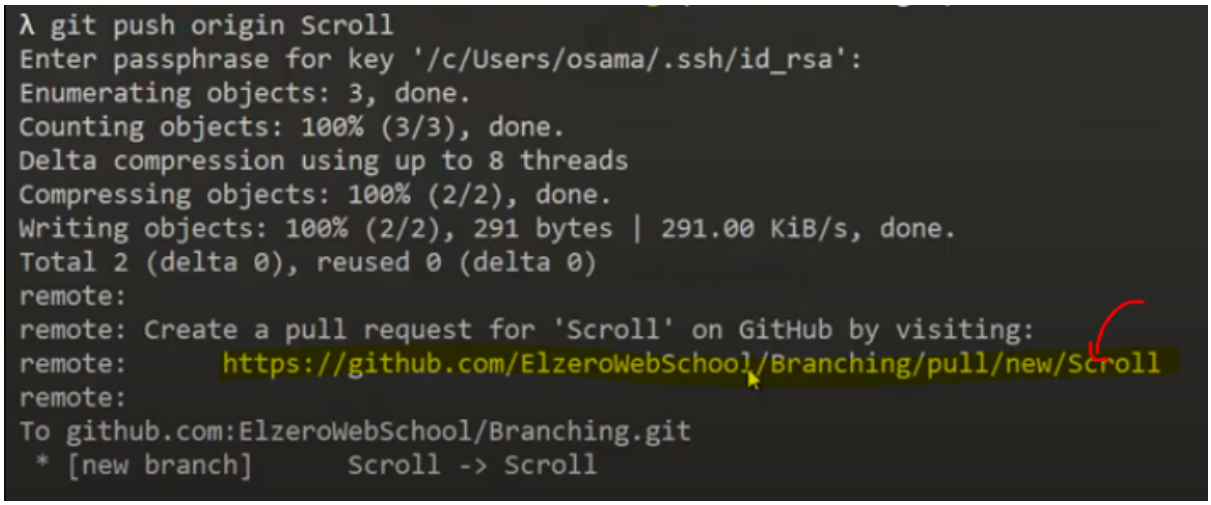
- To rename a branch

```
git branch -m ModifiedNameBranch
```

- After we created our branch we added a file inside. Lets say that I want to merge it with the main branch

```
#first go back to main
git checkout main
git merge BRANCHNAME
```

- What if we want to send our branch to our Github repo so that we can make a pull request. After we pushed our branch to our remote repo, it sent back a message to create a pull request on the link I highlighted



#13 Stash Part 1

- We can reserve files aside that I don't want to push to my repo in the mean time using the 'stash' command
- After you add your files to the staging area you can reserve them in an invisible git box called 'stash'

Example:

```
#create a folder called Hello which contains 'Hello world'
echo "Hello world" > Hello.txt
git add Hello
git stash
```

- Now ‘Hello’ will not appear in the working directory
- To make sure that our stash is now reserved in the stash box

```
git stash list
```

- To get out stashed files from stash

```
git stash pop
```

- Now our files will appear again in our directory, and all you have to do is to commit your changes so you can push them on the remote repo

#14 Stash Part 2

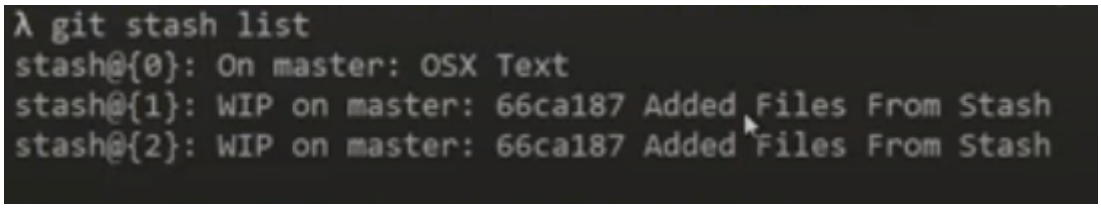
- If I want to add a file to the stash with a message so I can have an idea about the stash I added when I check the stash list

```
touch osx.txt
git add osx.txt
git stash save "OSX text"
```

- I can get a file from the stash without removing it

```
git stash apply
```

- I can get a specific stash from its id



```
#to remove stash 2
git stash pop stash@{2}
```

- To drop the last stash

```
git stash drop
```

- To drop a specific stash

```
git stash drop stash@{STASHNUMBER}
```

- To know what is the content of a stash, for example the first one

```
git stash show stash@{0}
```

- TO CLEAR ALL STASHES

```
git stash list
```

#15 - Restore And Clean

- To take back a file you added in the staging area, you use ‘git restore’ command

```
touch  new.txt
git add new.txt
git restore --staged new.txt
#Now you file is not staged again
```

- If you have more than one file and you want to unstage all files

```
git restore --staged *
```

- To check what files you want to remove from the unstaged area

```
#It will tell you what are the files that would be removed from the staging area
git clean -n
```

- To remove unstaged files

```
git clean -f
```