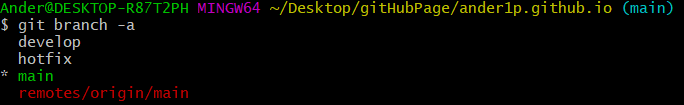
# Branching Strategy

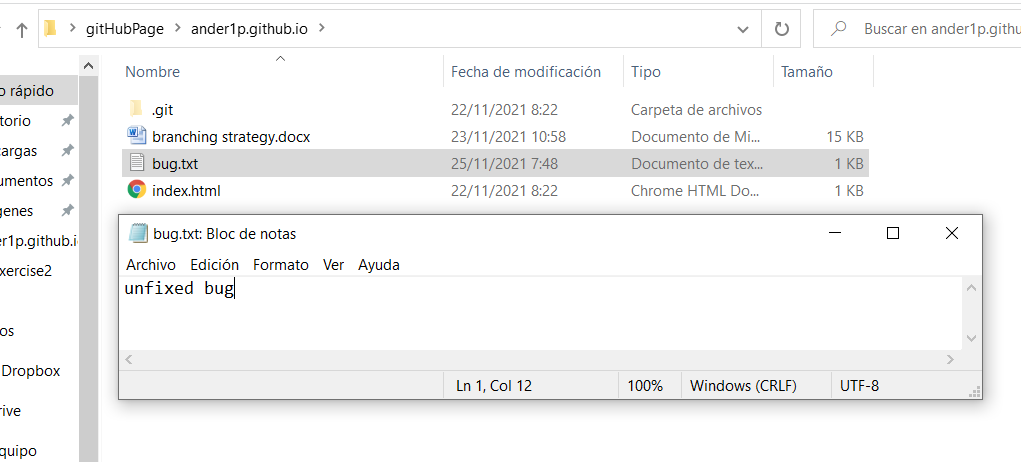
To create this webpage I will use three branches, the master branch, where stable versions will be uploaded, develop branch, where new features will be added and once they are working will be merged to the master branch, and a hotfix branch, where I will correct the errors I notice in the web page and them merge them to master.

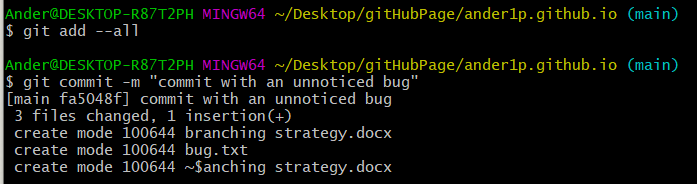


# The commands used are:

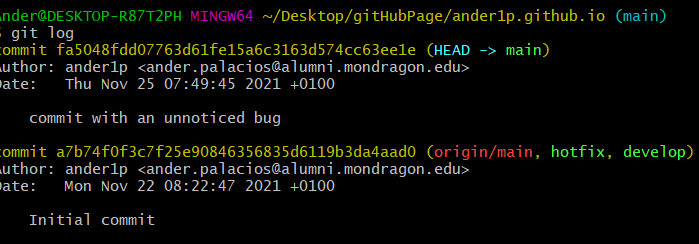
## git revert

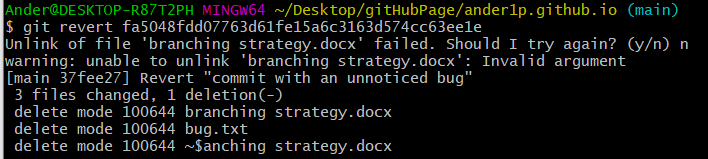
After committing an unwanted document, like bug.txt we can revert the files so that we can continue developing without the bug.



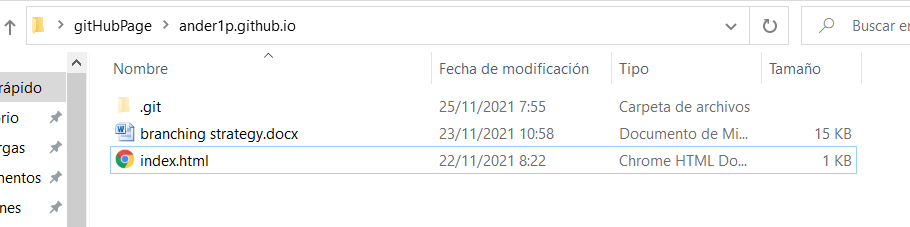


We will revert this commit to eliminate the bug finding its hash with log command



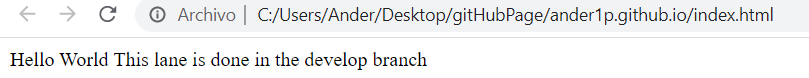


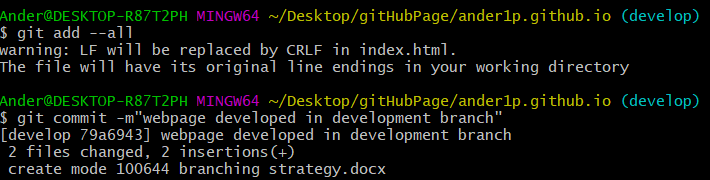
The .txt has been deleted



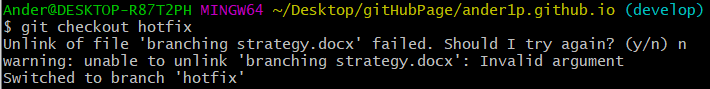
## git rebase

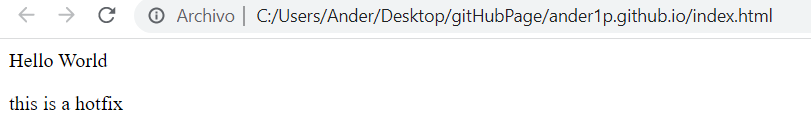
First we will put HEAD pointing at develop branch, update the web page and commit our changes

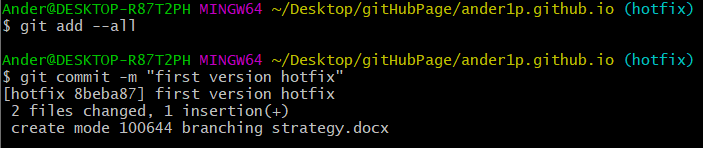




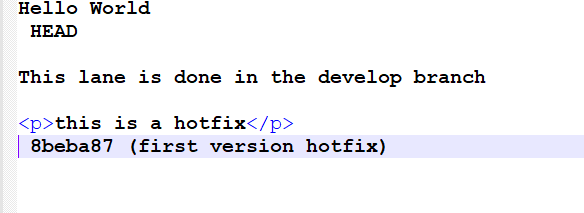
Then we will point at hotfix branch and commit a hotfix



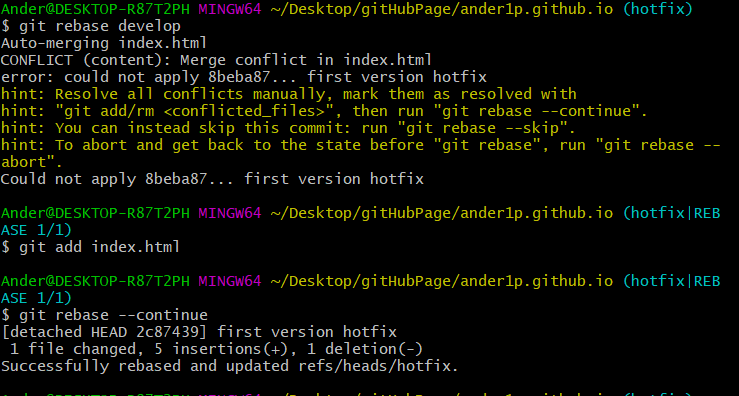


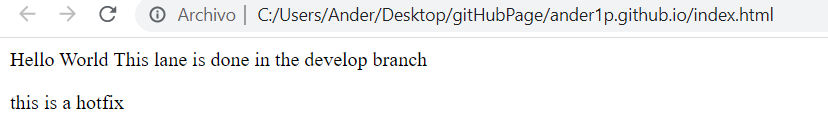


Now we will rebase the previous commit of the develop branch, while doing this conflicts must be solved by editing the document



Then we will add the conflicted file and we will see that the two branches have combined successfully.

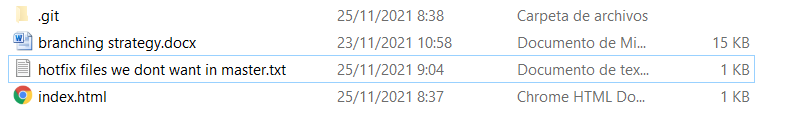


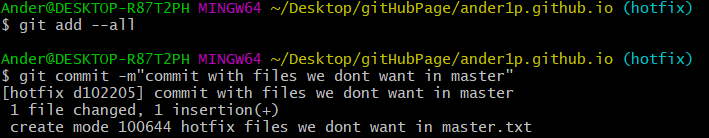


## git cherry-pick

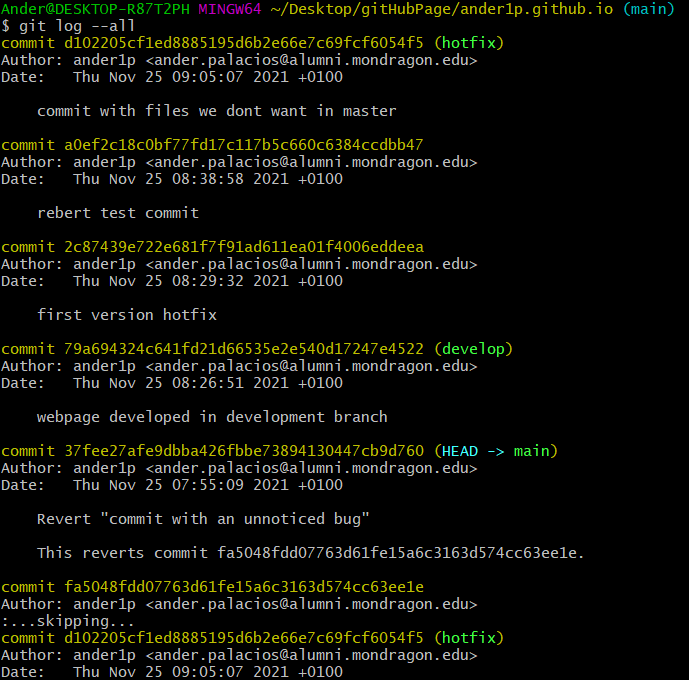
To do the cherry-pick we will use the commit done in the previous point ant merge it to master

To use the best part of cherry-pick we will commit in hotfix a version that we don’t want to merge to master, and select the one where we used rebase.

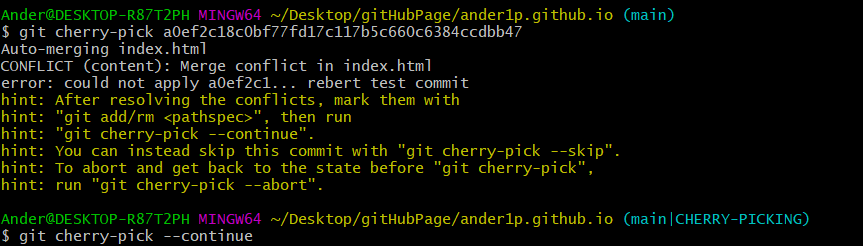


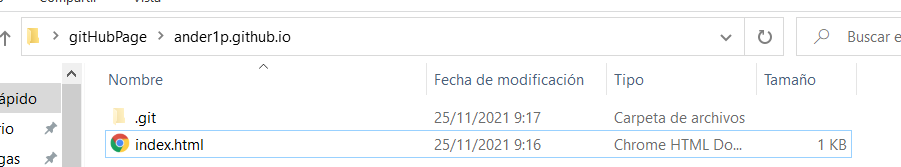


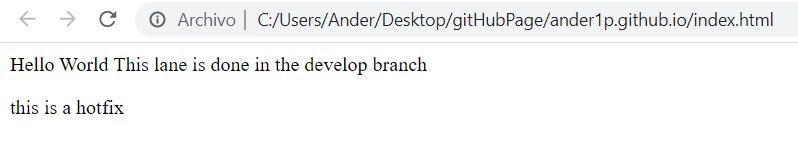
Then we will checkout main and check all the log with git log --all to find the hash of the commit we want



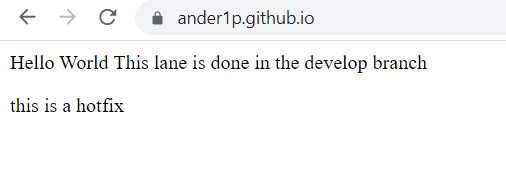
Once we have it we will use the cherry pick command with it to find it only has updated the index but not created the unwanted files





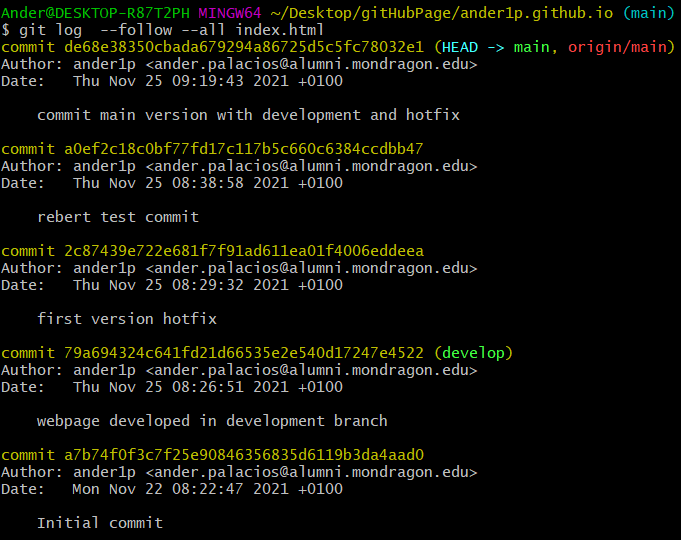


And finally commit all our changes in main



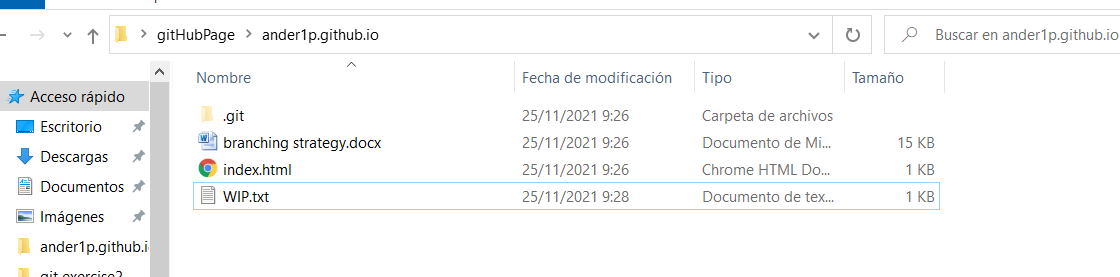
## git log --follow --all [file]

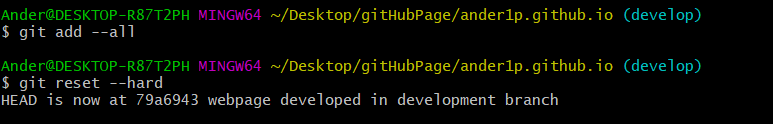
With this command we can follow the changes done to a certain file of our project, such as the index.html in all branches

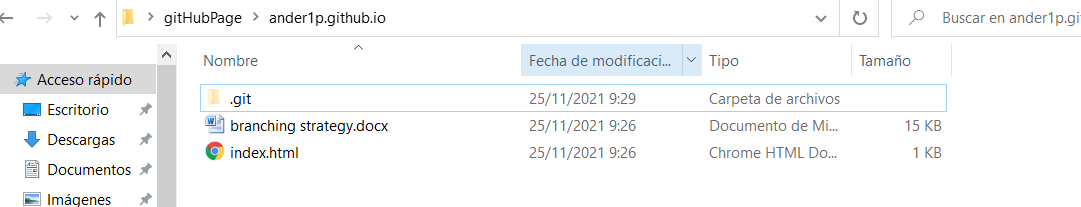


## git reset --hard

This command is used to delete all the work in progress in the specific branch, to test it we will use it in the develop branch after creating a new file there





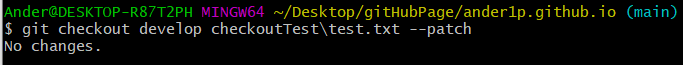


As we can see the created document has been deleted, and we have the exact documents we had in the initial commit

## git checkout <branch name> <path to file> --patch

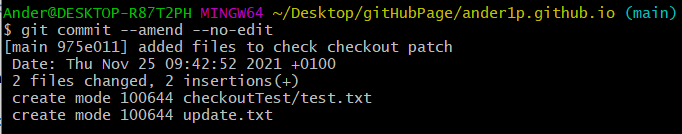
This command is used to restore paths to files we don’t have in a certain branch, in this case I have created a folder to test where y created a certain document, in the main branch, later in develop I merged this changes and changed the route of the folder I commit it and then go back to main to update the route, with the following outcome.





## git commit --amend --no-edit

Just after a commit I found that a file is missing, in order to not do another commit with another message I can use this command to update the previous commit



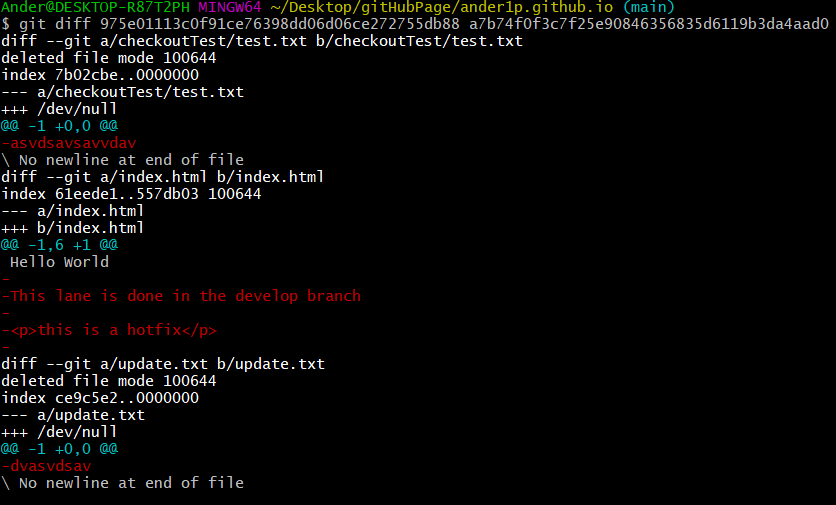
## git grep

This command can be used to find a certain word or function in the current branch



## git diff <sha1> <sha2>

With this command you can see all the differences between commits.



Author: Ander Palacios

URL: https://ander1p.github.io/