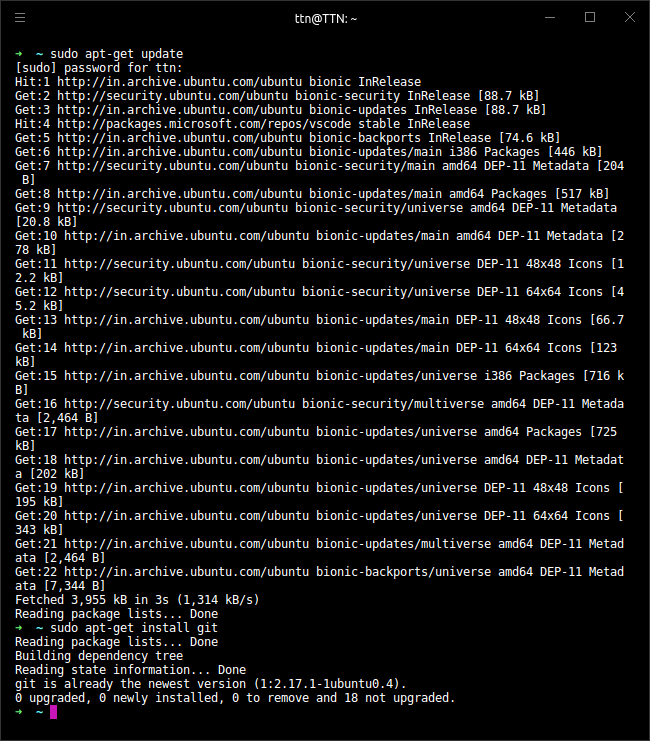
**Exercise**

**Introduction To Version Control**

**Name: Mahesh Inder**

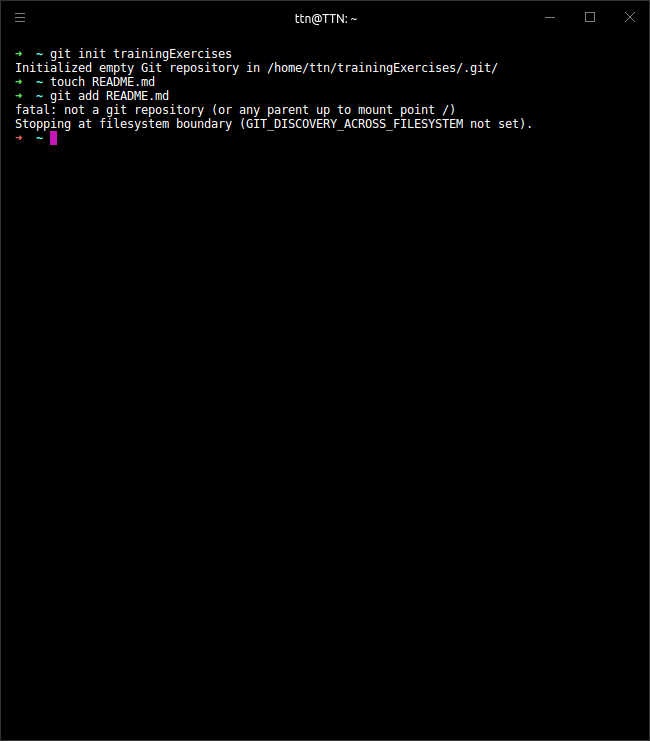
1. **Git Setup** [**https://confluence.atlassian.com/bitbucket/set-up-git-744723531.htm**](https://confluence.atlassian.com/bitbucket/set-up-git-744723531.html)
2. Instaling git:

* sudo apt-get update
* sudo apt-get install git

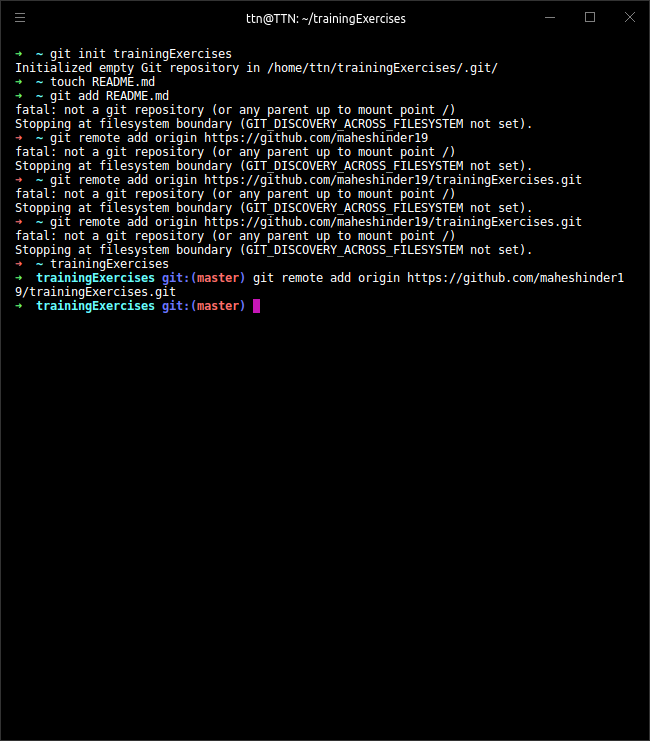


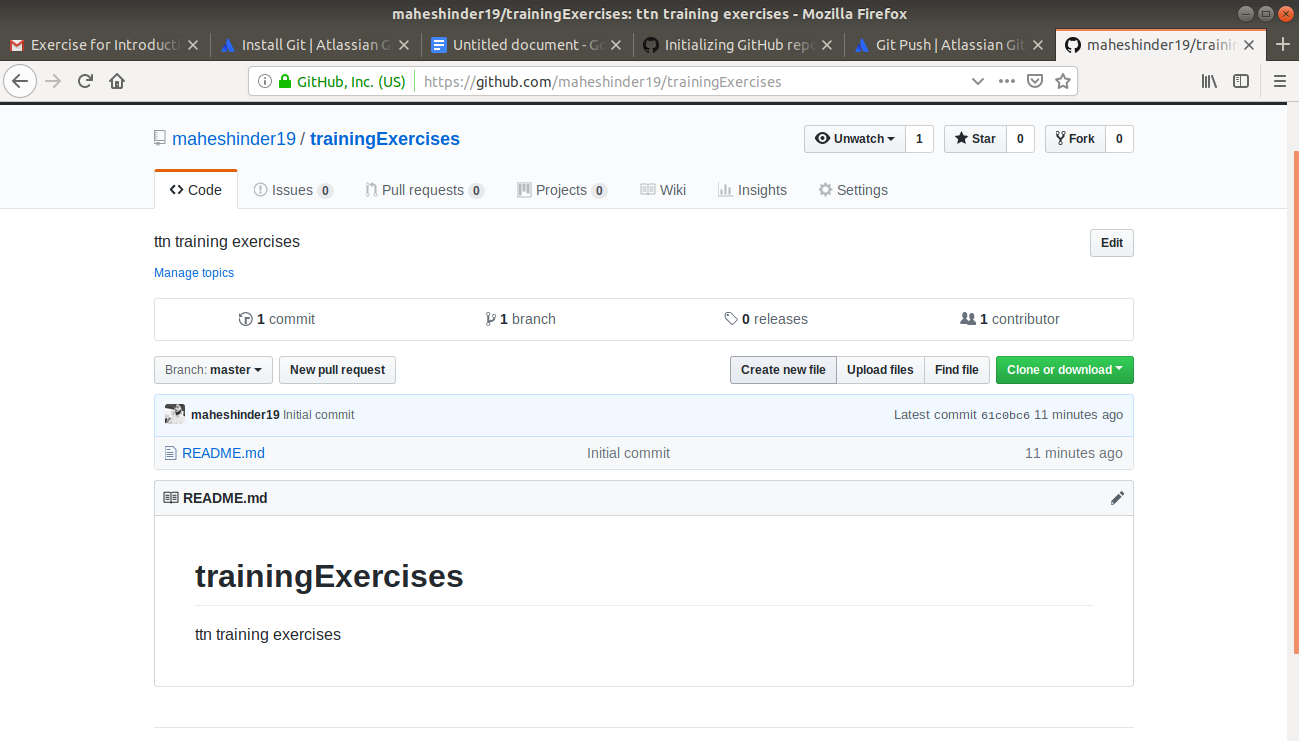
**2. Initialize a Git Repository.**

* We can initialise a git repo by using “git init” command from the terminal.

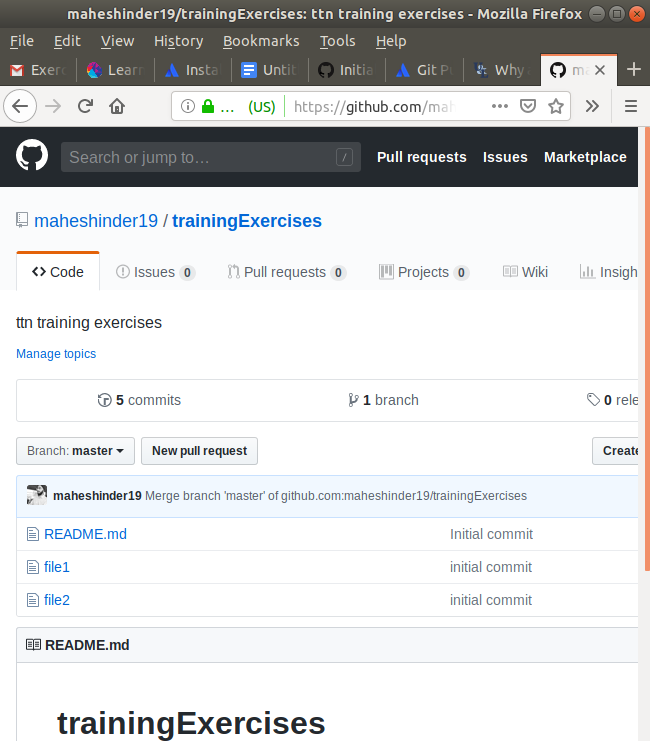


* Now by using “git remote” command, route the repository to the desired remote repository.



* Check the online repository.

**3. Add files to the repository.**



**3. Unstage 1 file**

1. A file can be unstaged by using the command “git --cached fileName”



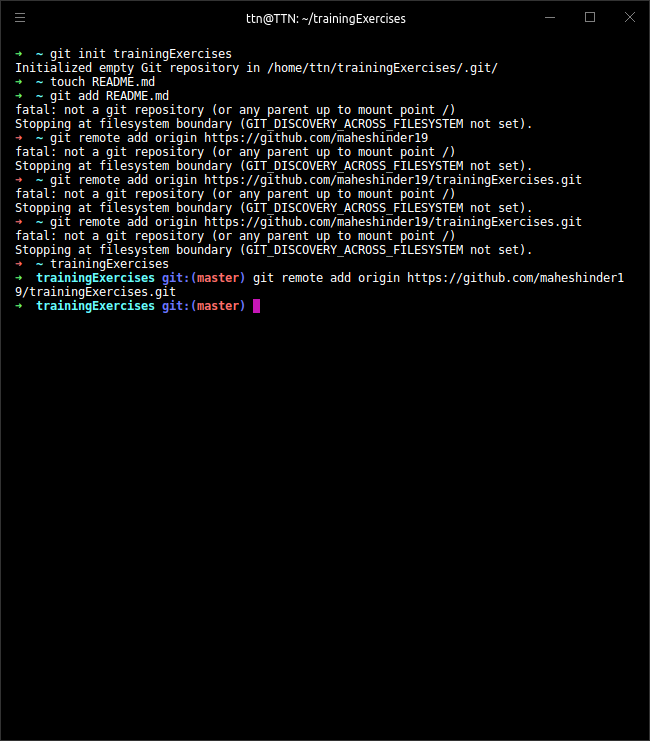
**4. Commit the file**

1. To commit a file, a file is staged first using “git add” command and then committed using “git commit”.



**5. Add a remote**

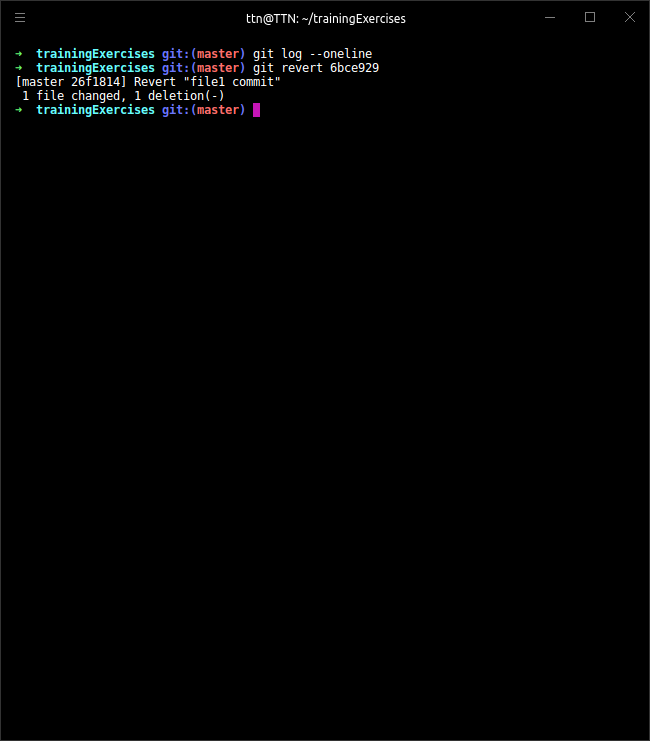
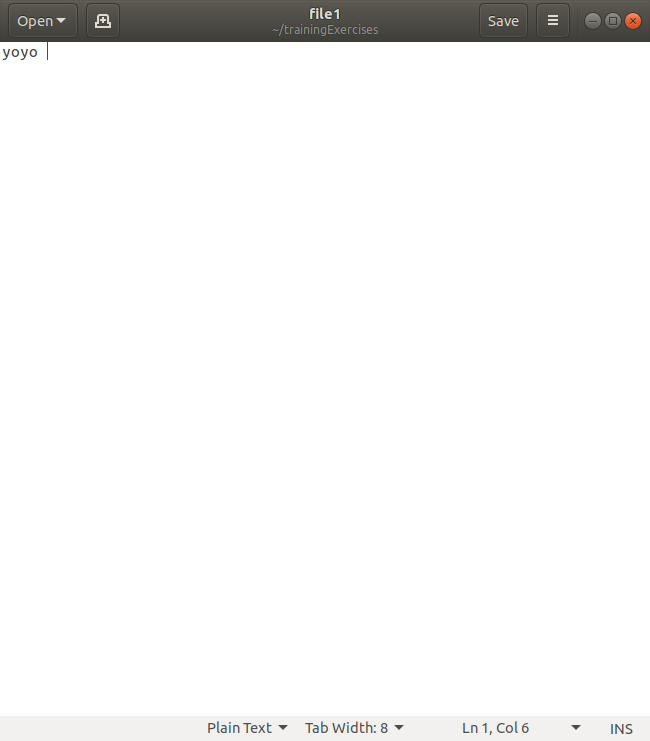
1. We can add remote by using the “git remote add/set-url” command followed by the url to the remote repo.



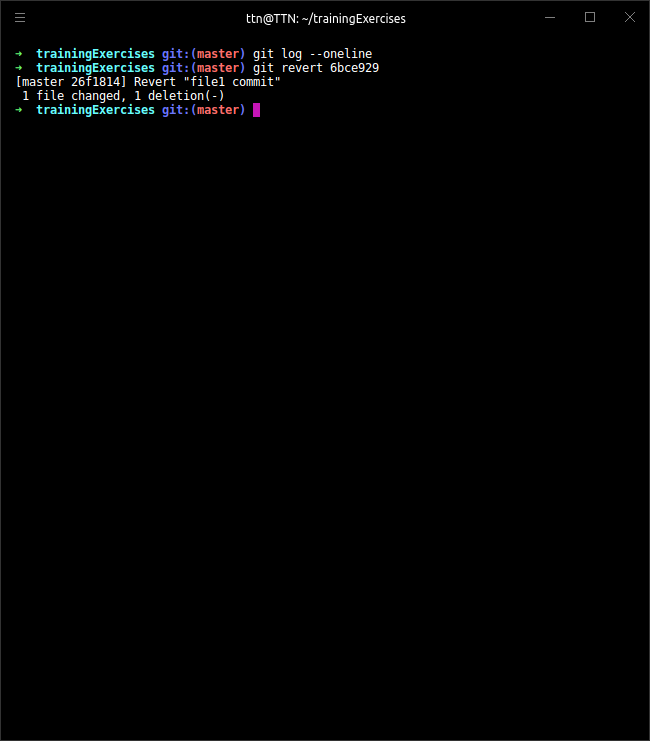
**6. Undo changes to a particular file.**

1. Changes in a file can be undone by using “revert” command as shown below:

Suppose we have a file named “file1” with the last commit as follows.

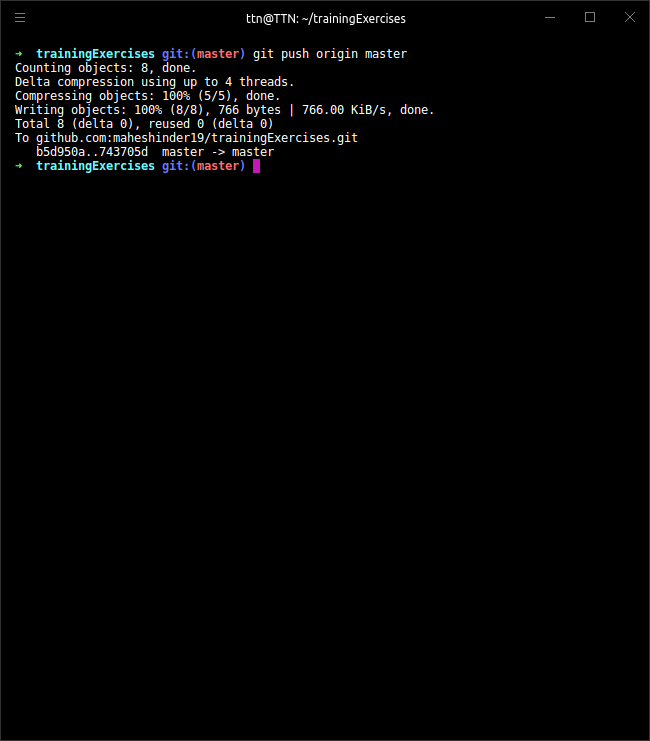
We need to find the commit hashcode for the above commit to undo it.

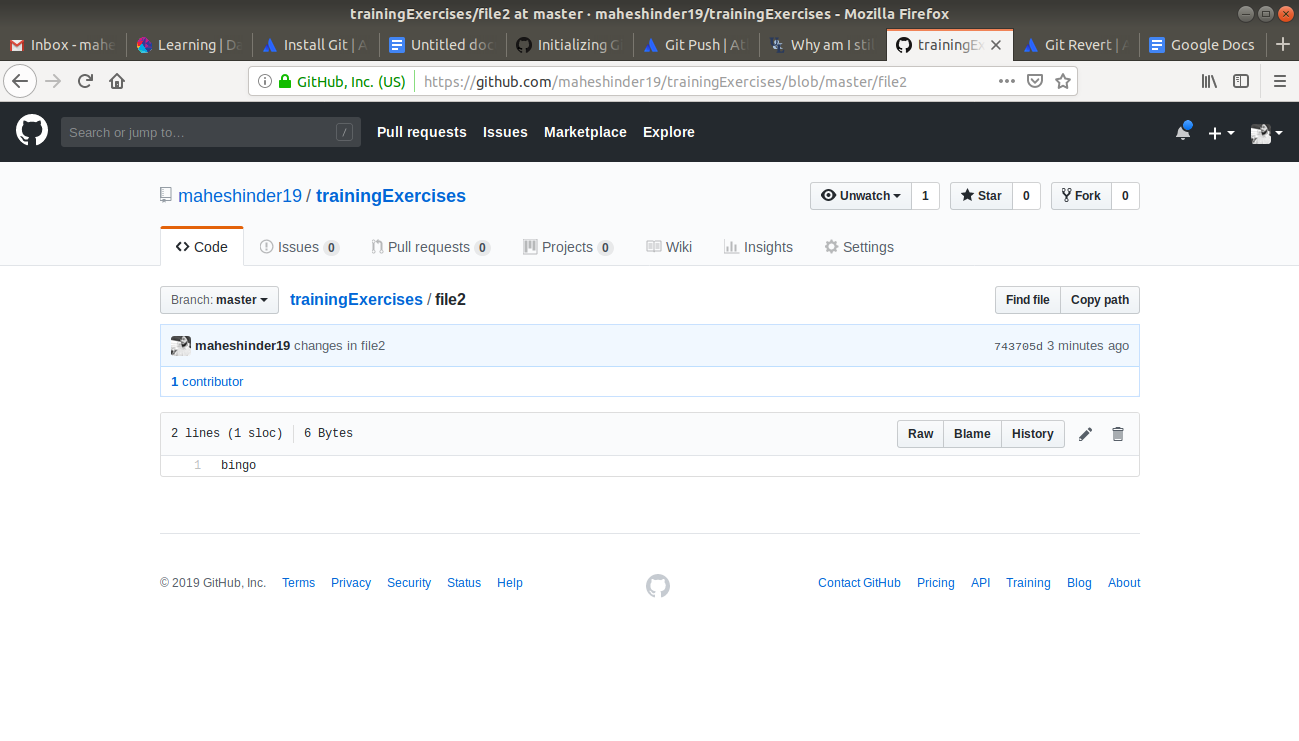
By looking at the log we can get the informations about the commits.

Now, to undo the changes of the commit “6bce929”, we need to use “revert”.

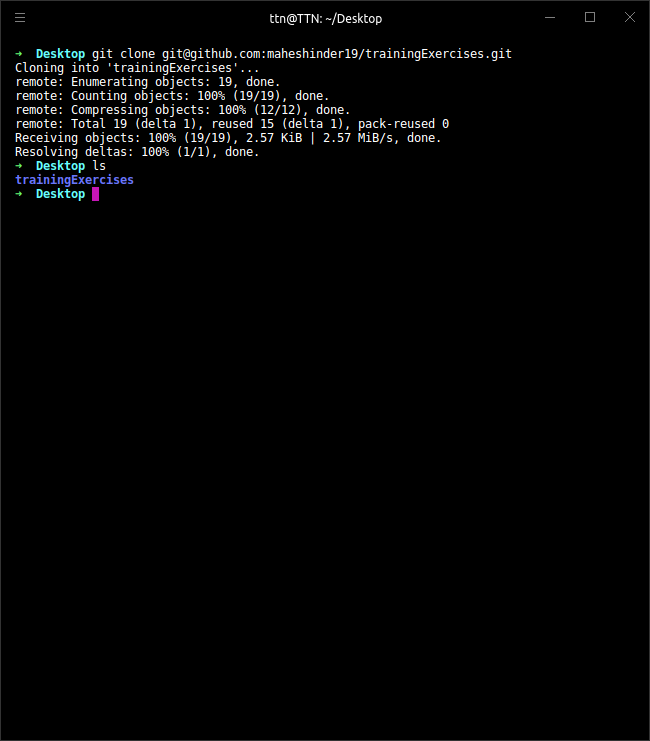
Changes get reflected on the file “file1”(text “yoyo” deleted).

**7. Push changes to Github**

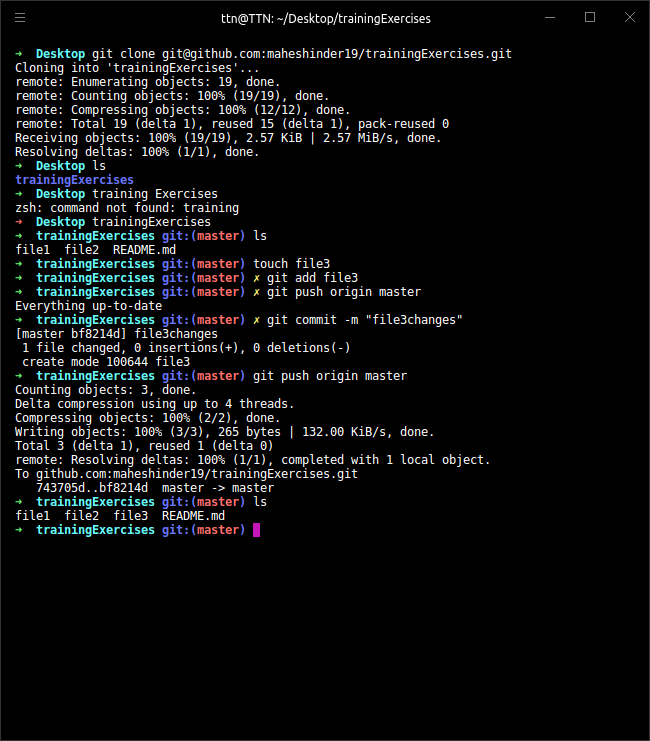
* Making changes in “file2”
* Commiting changes and pushing to Github repo.
* Push successful.

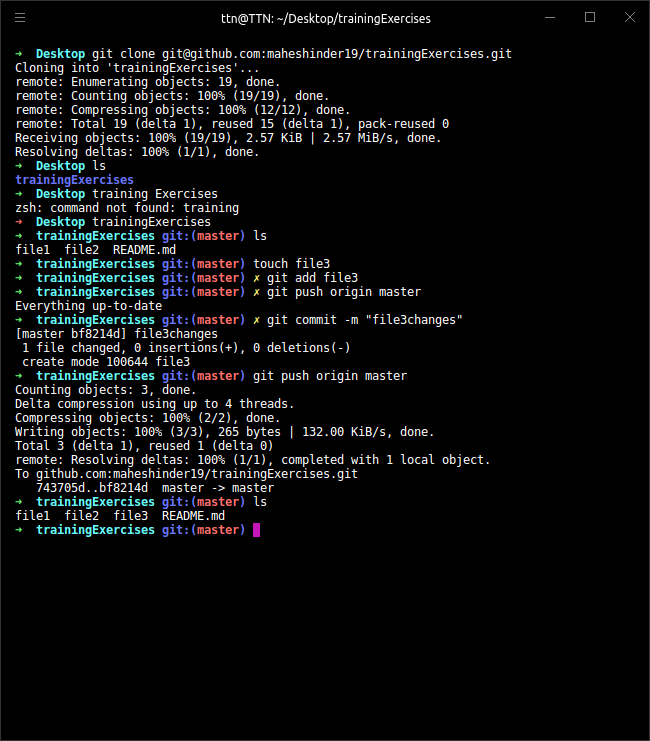


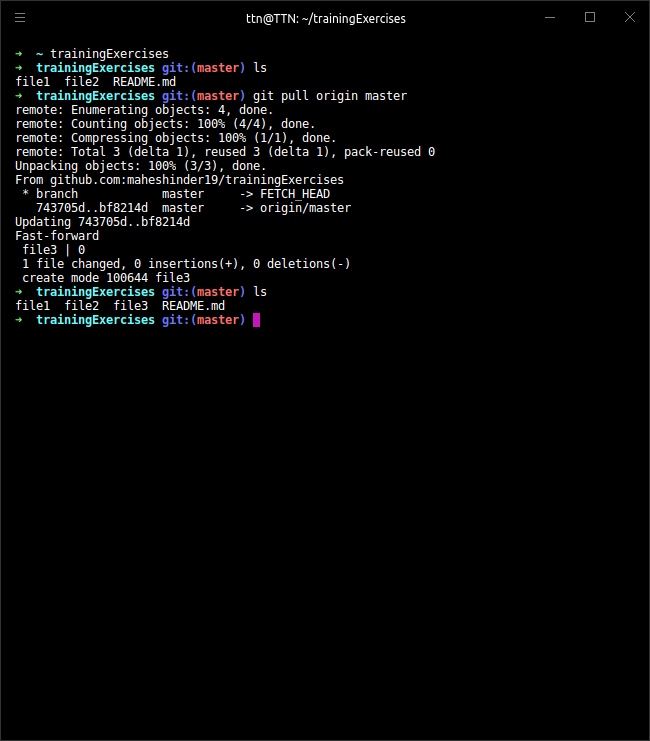
**8. Clone the repository**

1. One can clone any repository by using “git clone” command followed by the url of the git repo.

**9. Add changes to one of the copies and pull the changes in the other.**

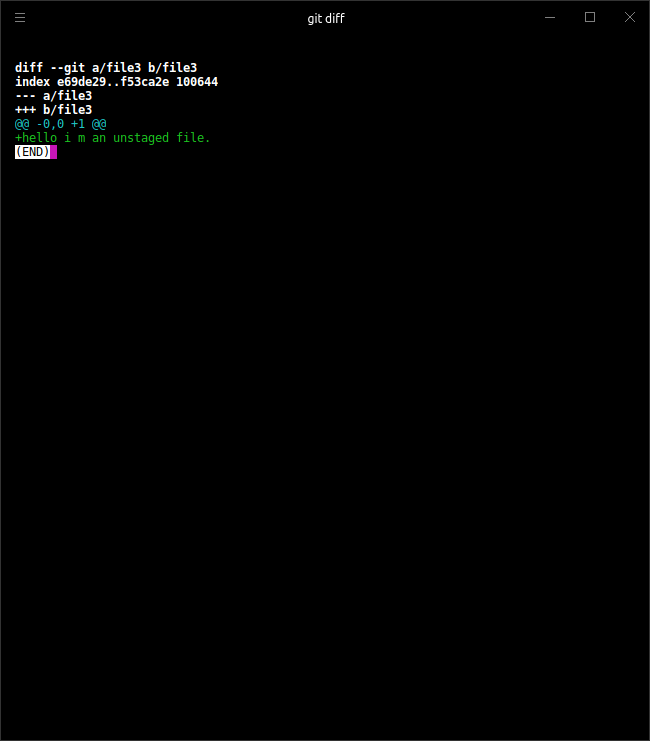
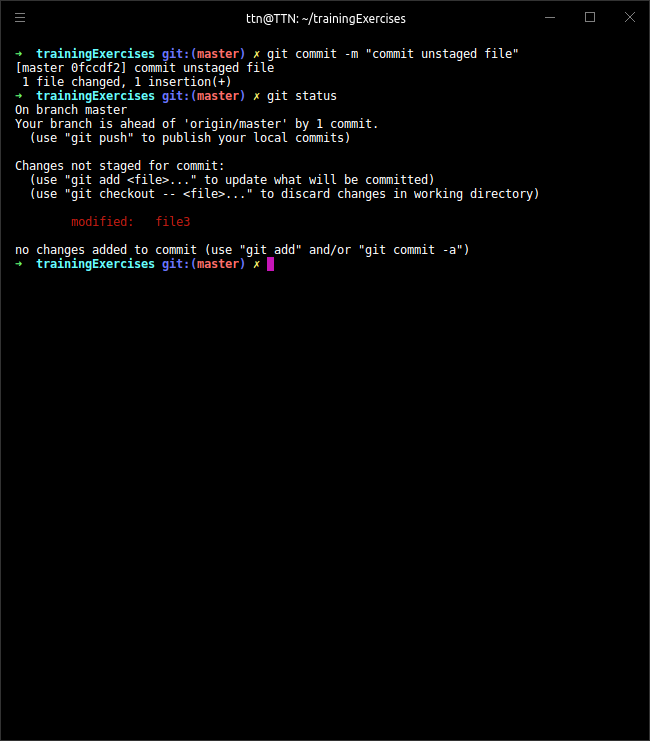
* Making changes in the copy of repo in Desktop and pushing the changes.



* Changing working repo to the original one and pulling the changes. 

**10. Check differences between a file and its staged version**

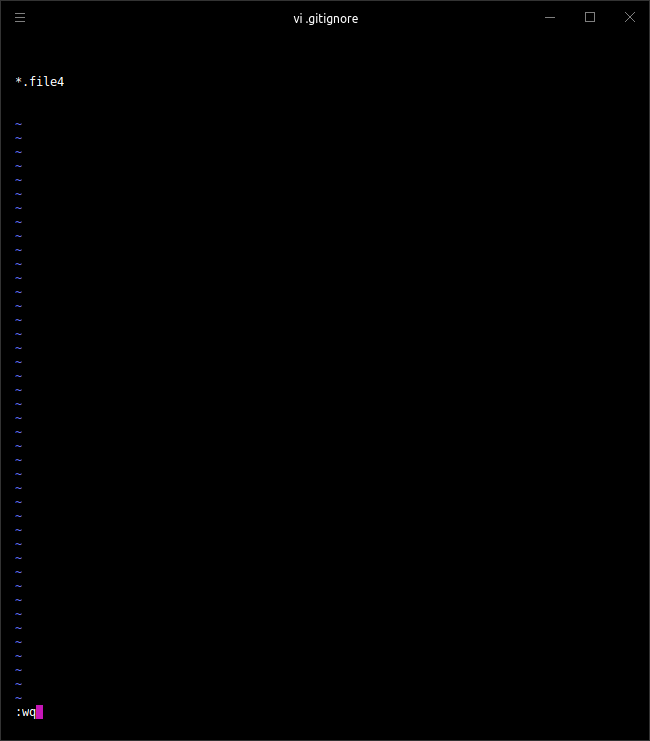
1. I have made some in changes in file and didn’t stage it. Let’s see the differences between the staged and unstaged version by using “git diff” command.

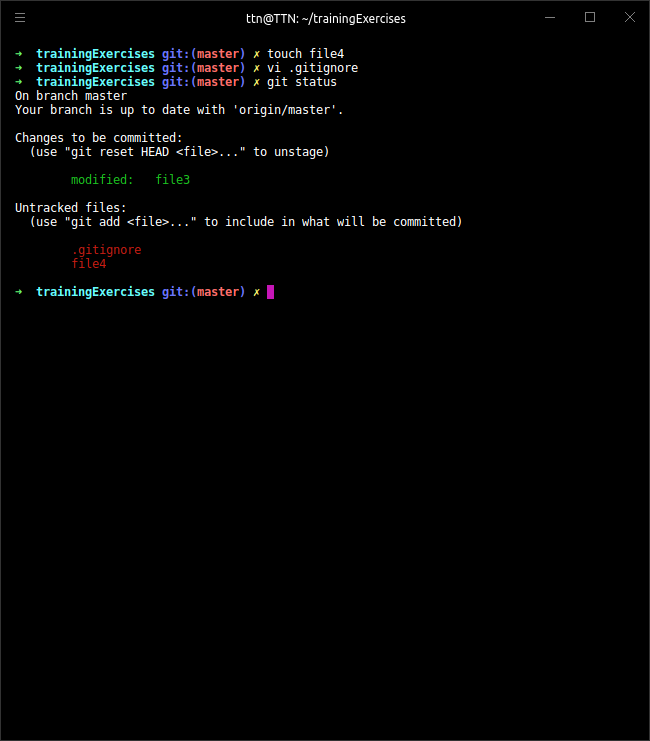
Staging prepares a file to get committed. If a file is unstaged, no changes will reflect in it after the commit.

11. Ignore a few files to be checked in.

1. A file named “gitignore” is used to specify the names of the files that need not to be tracked by Git.

Suppose we want file4 to be untracked.

Now adding file4 to “gitignore” file.

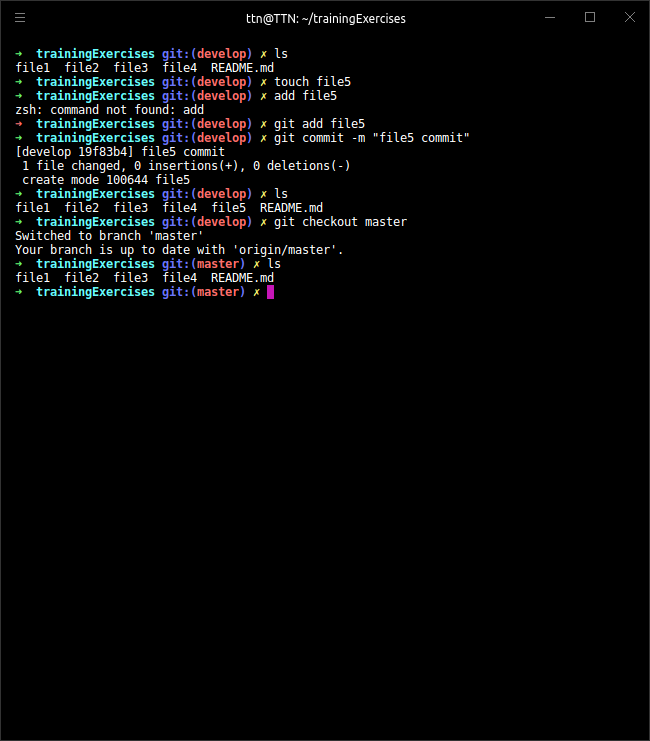
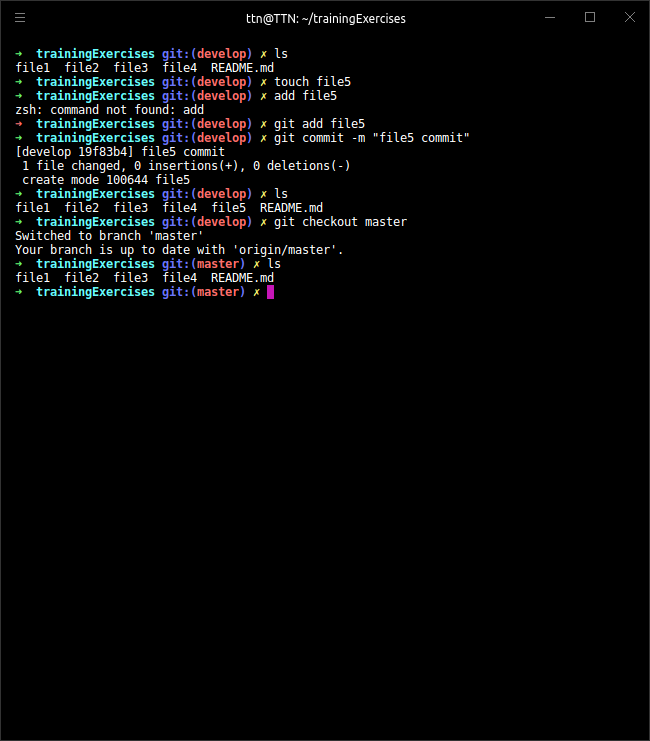
Checking git status.

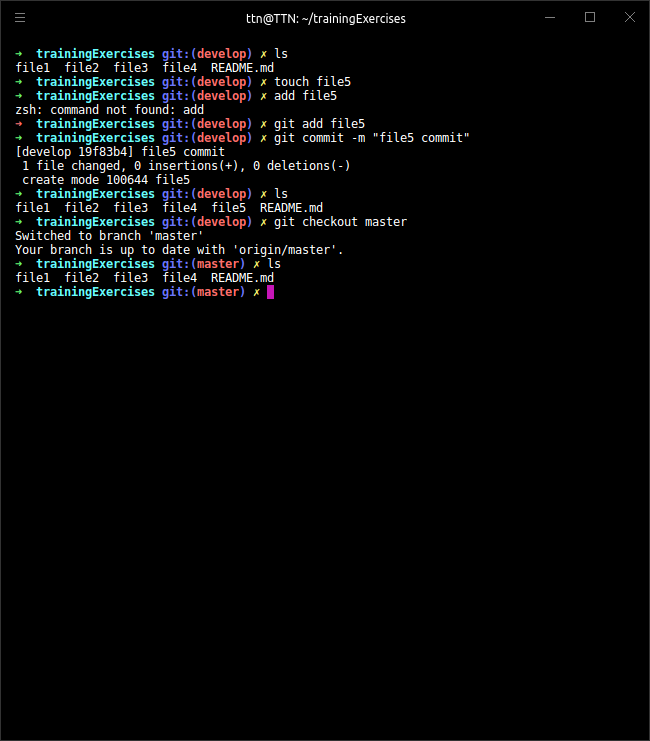
**12. Create a new branch.**

1. A new branch can be created by the following command:

“git checkout -b branchName”

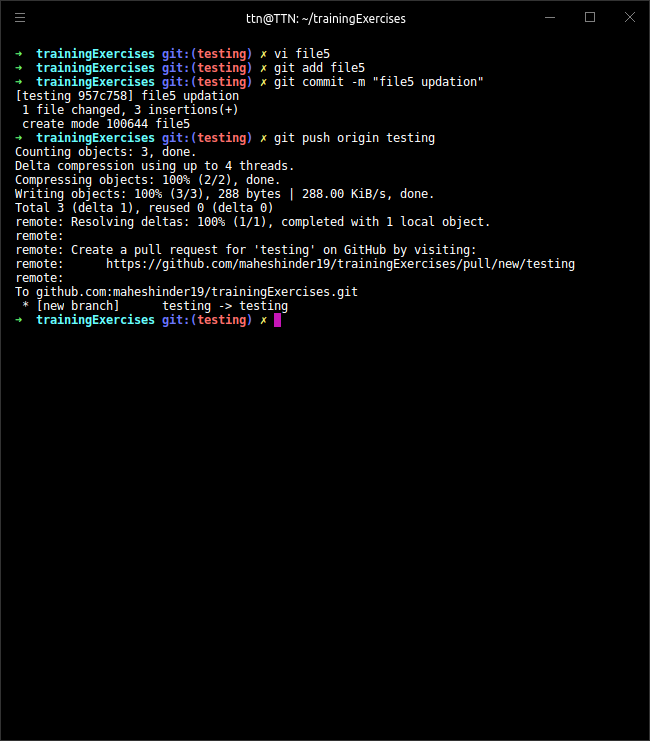
**13. Diverge them with commits.**

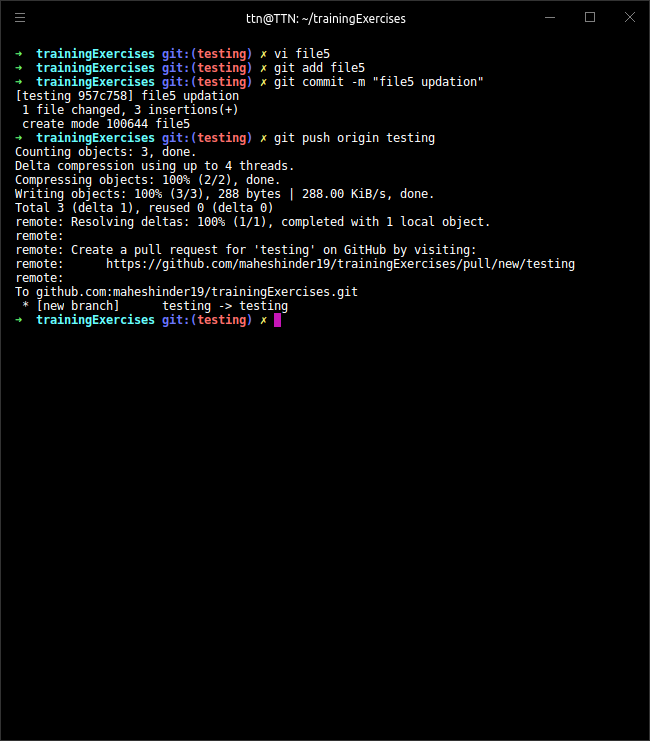
1. Committing changes in branch “develop” (adding “file5”) 

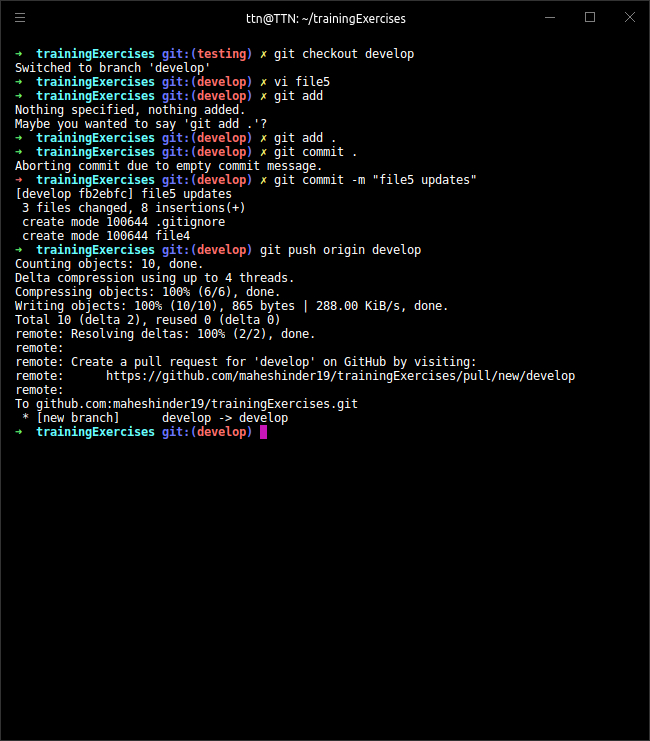
Checking “master” branch:

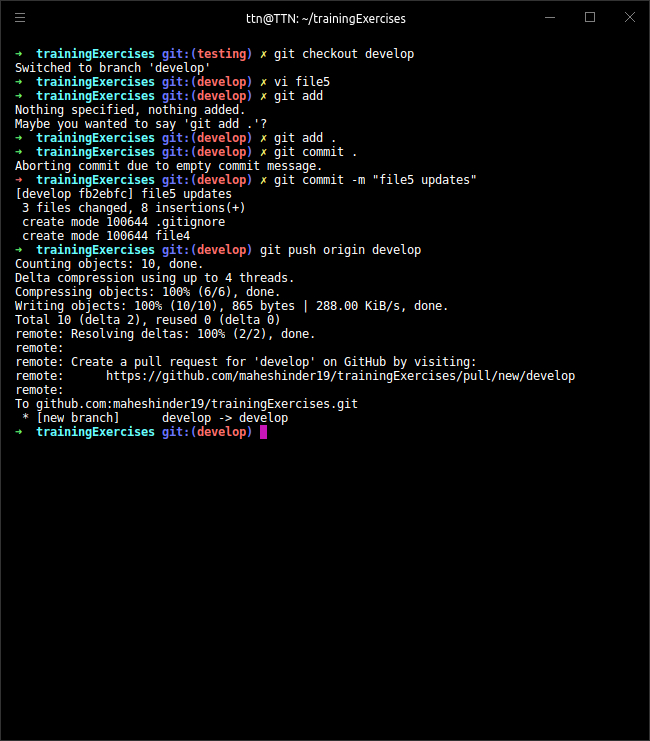
As we can see both the branches differ.

**14. Edit the same file at the same line on both branches and commit.**

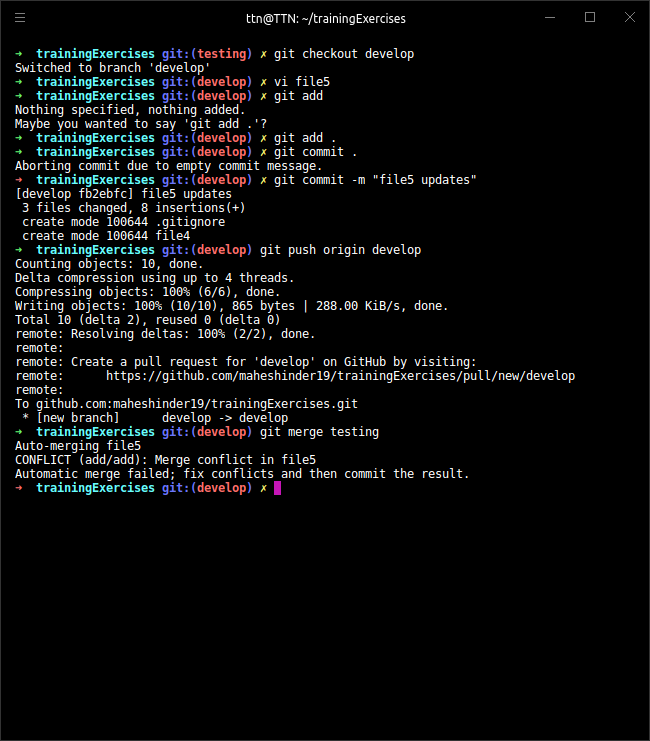
* Editing file5 in testing branch.

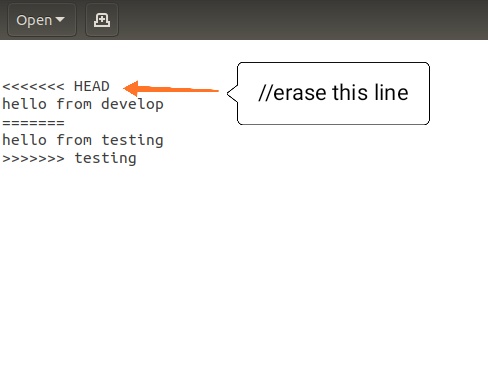


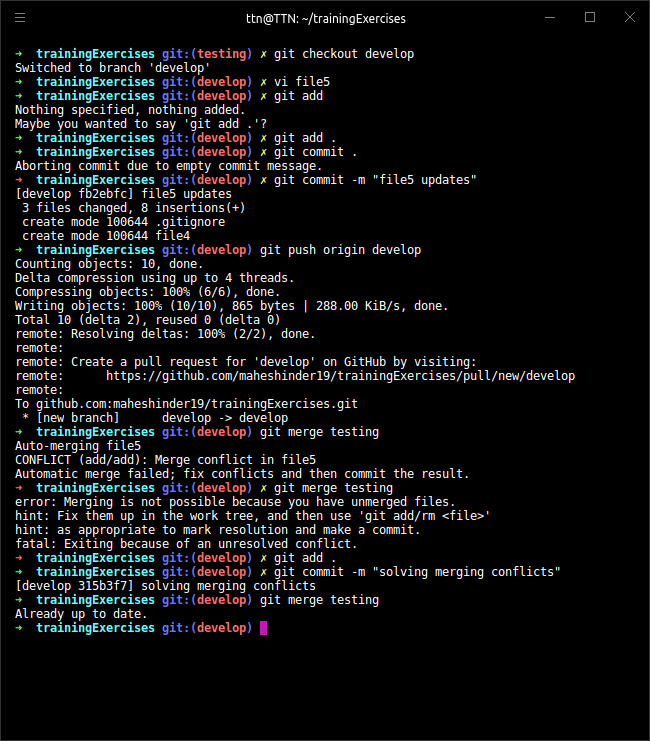
* Editing file5 in develop



**15. Try merging and resolve merge conflicts.**

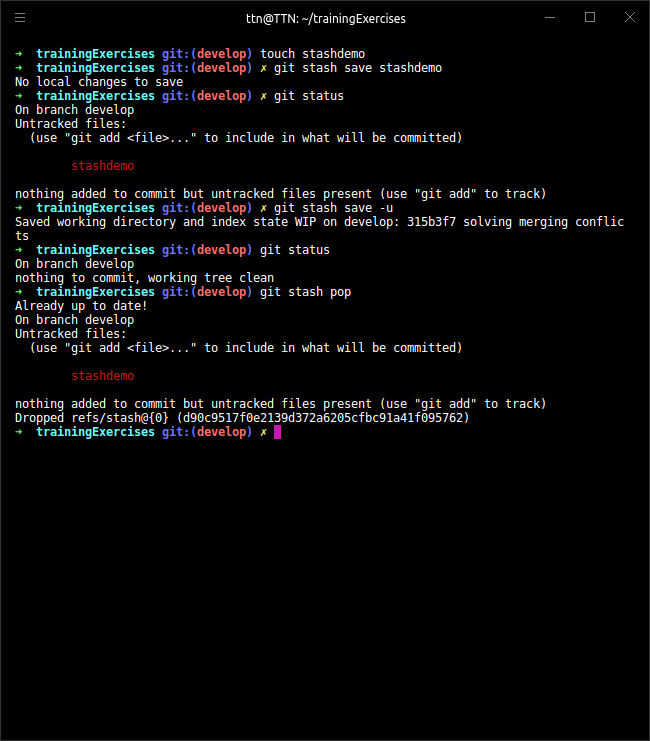
1. On merging develop and testing branches we get conflicts as the file called “file5” has same text on same line.

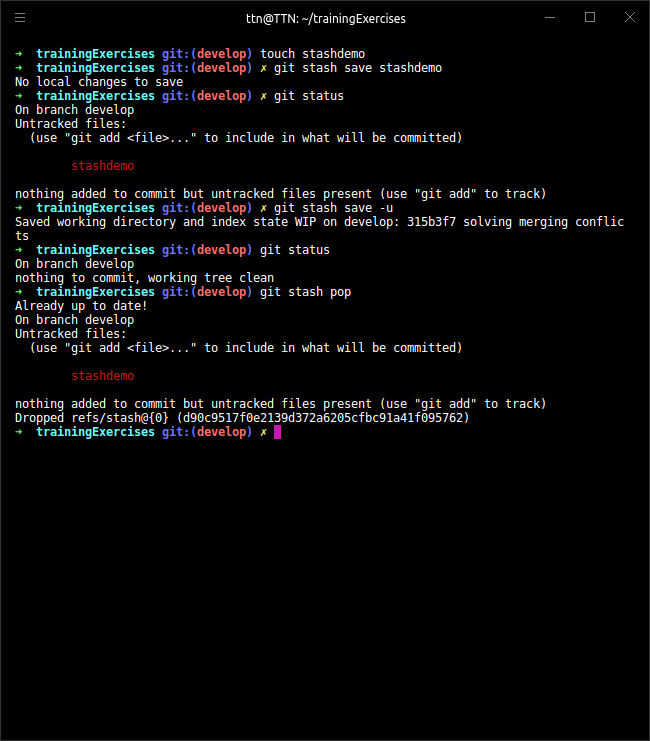
To resolve the conflicts we need to look at the “file5” and erase the “HEAD”.

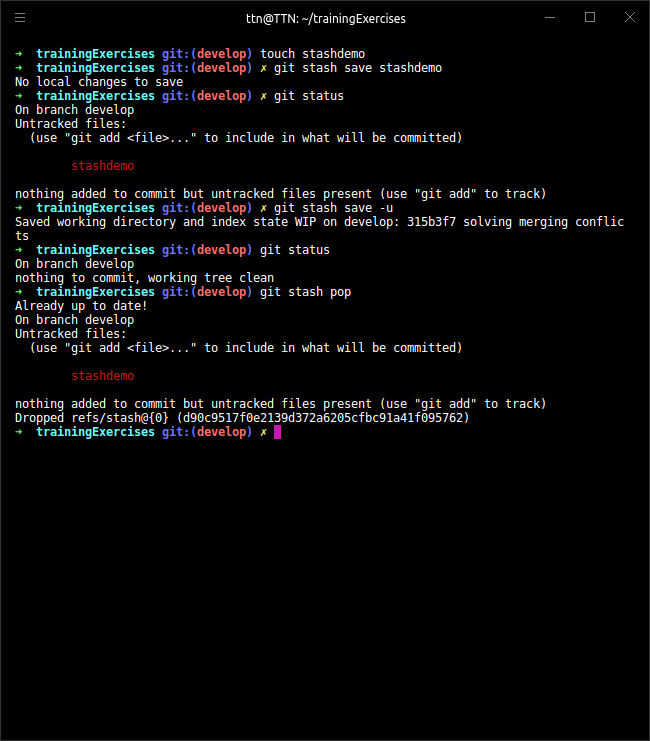
Pushing the changes and check if the conflict is resolved.

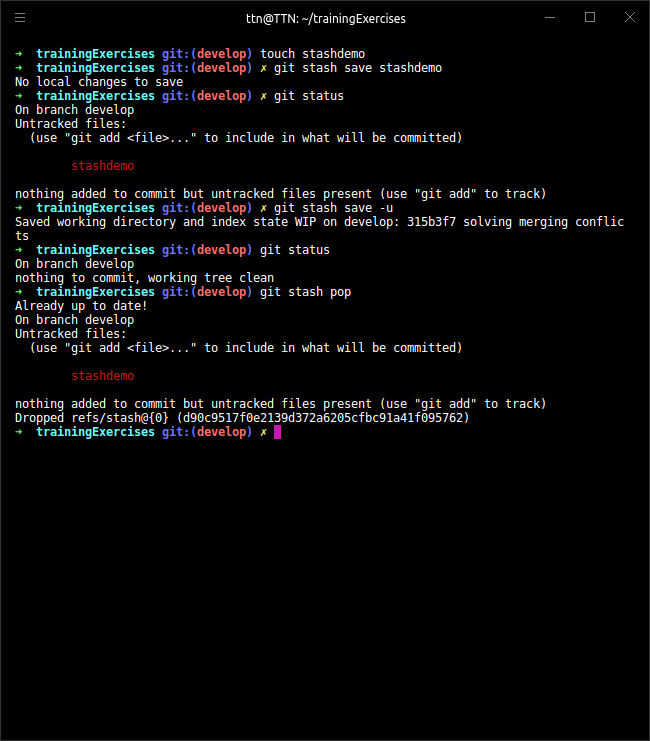
**16. Stash the changes and pop them.**

1. We can stash the changed if we do not want to commit them and do not want them to create any interruption in pushing.

* Creating a file “stashdemo”.

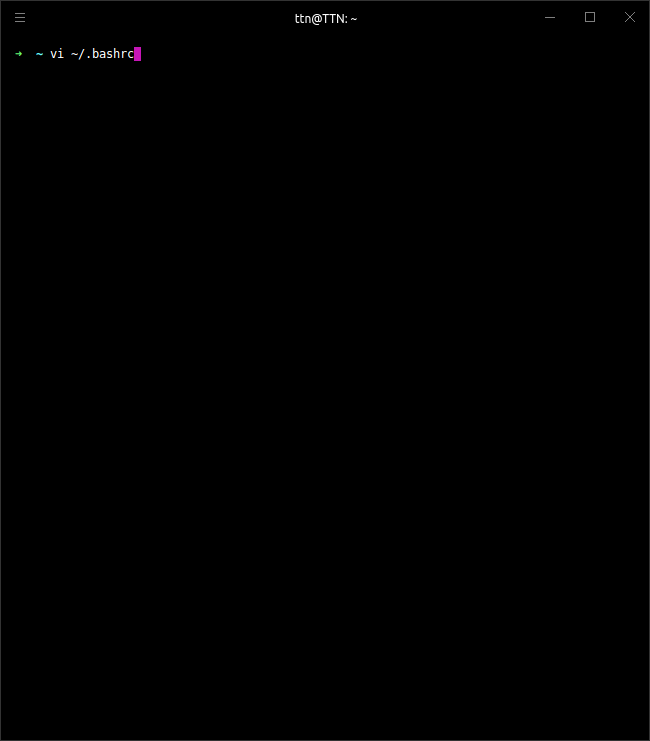


As we can see “stashdemo” is unstaged. Let's save it in stash.

* Now let’s pop it.

File “stashdemo” is again unstaged for changes to be made.

**17. Add the following code to your .bashrc file : color\_prompt="yes"  
 parse\_git\_branch() {  
 git branch 2> /dev/null | sed -e '/^[^\*]/d' -e 's/\* \(.\*\)/(\1)/'  
 }  
 if [ "$color\_prompt" = yes ]; then  
 PS1='\u@\h\[\033[00m\]:\[\033[01;34m\]\W\[\033[01;31m\] $(parse\_git\_branch)\[\033[00m\]\$ '  
 else  
 PS1='\u@\h:\W $(parse\_git\_branch)\$ '  
 fi  
 unset color\_prompt force\_color\_prompt**

1. Opening bashrc using vi.

Adding code.

