Week 1 - Assignment

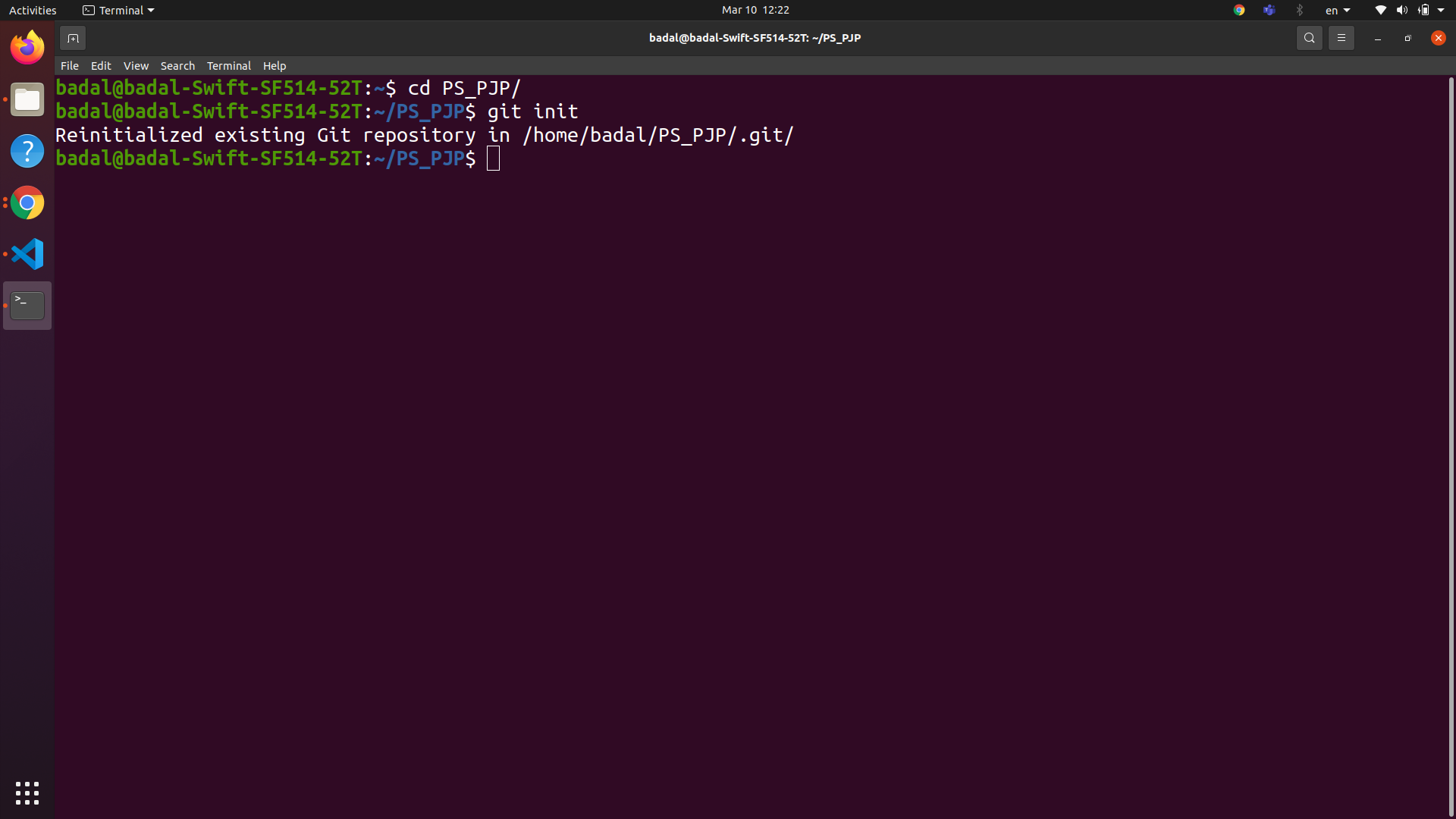
1. **Github**

* a

(i) Ans: Create a folder (eg: PS\_PJP)

Terminal command : cd PS\_PJP

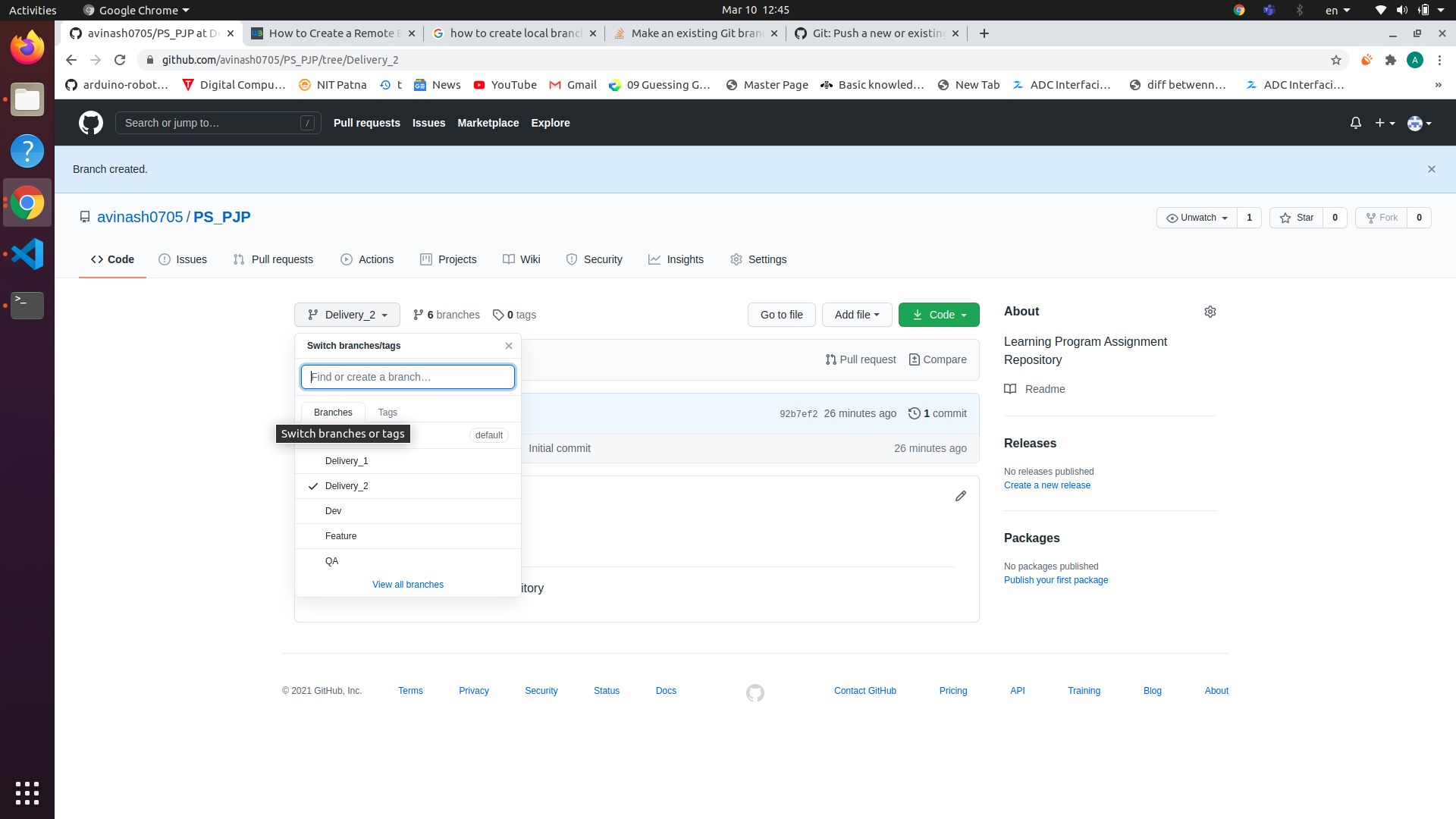
git init



(ii) remote repository :



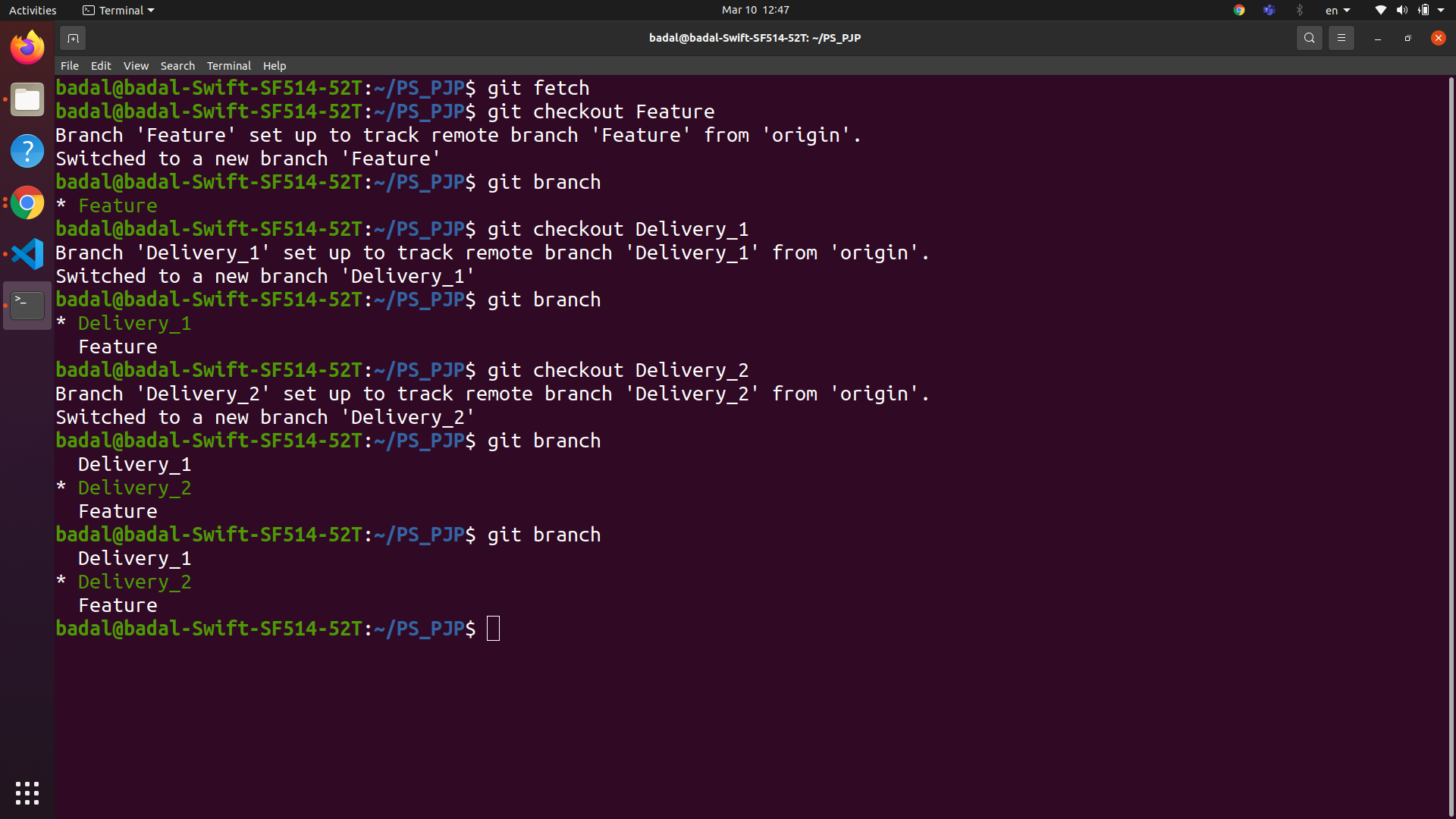
( iii ) remote branches



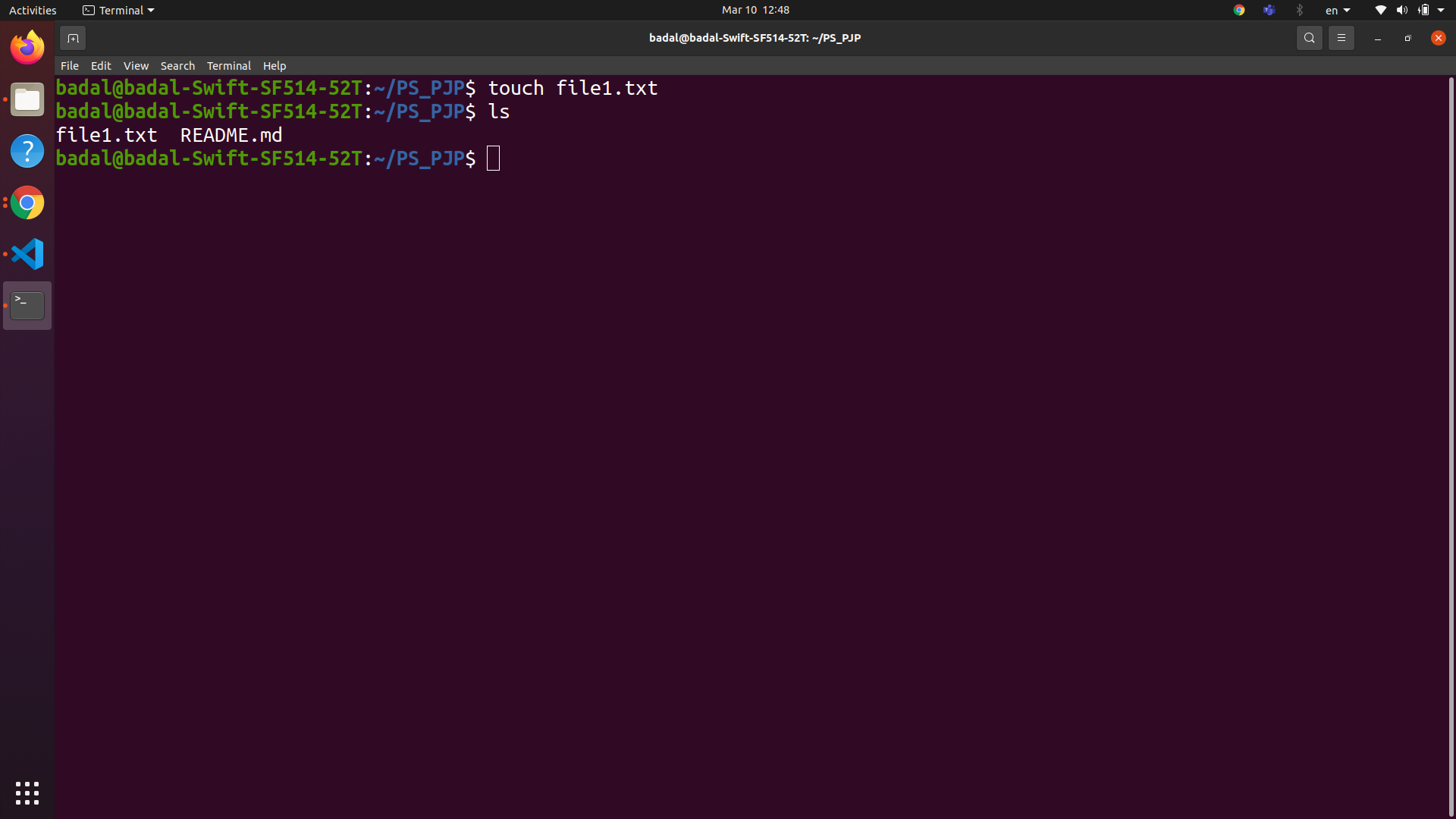
( iv ) local branches :

Checkout and remote branches in local and set the remote

Command :- git checkout “branch name”



( v) Add files



Stage and Commit those changes

git add .

git commit -m “message”

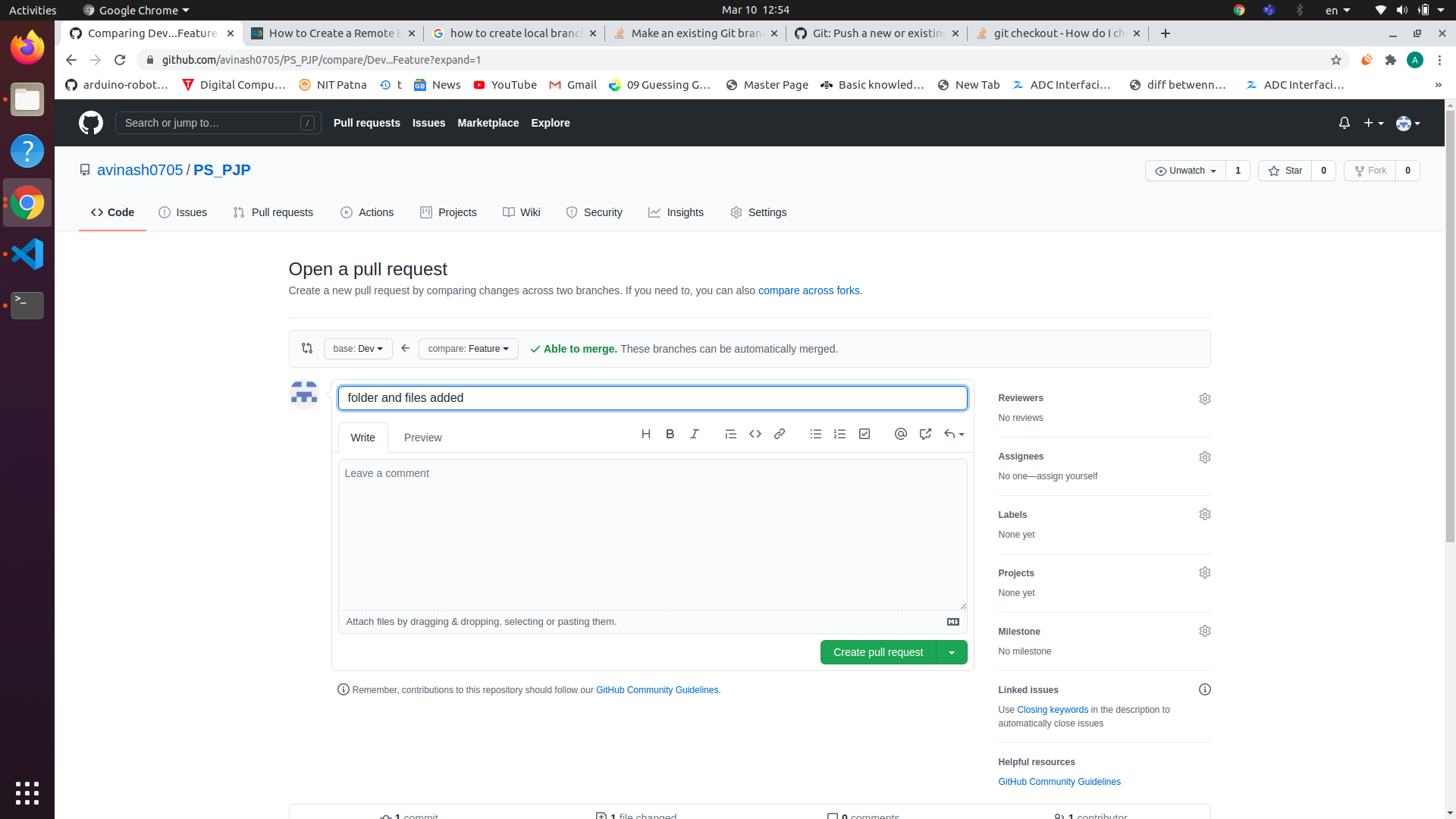


(vi)

Push files to feature branch

git push origin Feature

(vii) creating pull request



(viii)

Checkout latest code in local from remote

git pull origin “name of branch having latest code”

(ix)

**Checkout VS PULL**

Checkout is to move the head to a specific branch or commit (if checkout is done by commit id)

Pull means adding the changes to the local folders / repo that are present in the remote repository

(xi)

Sync code of all branches :

Checkout the branch in local

git checkout “branch\_name”

Pull the changes

git pull origin “name of branch”

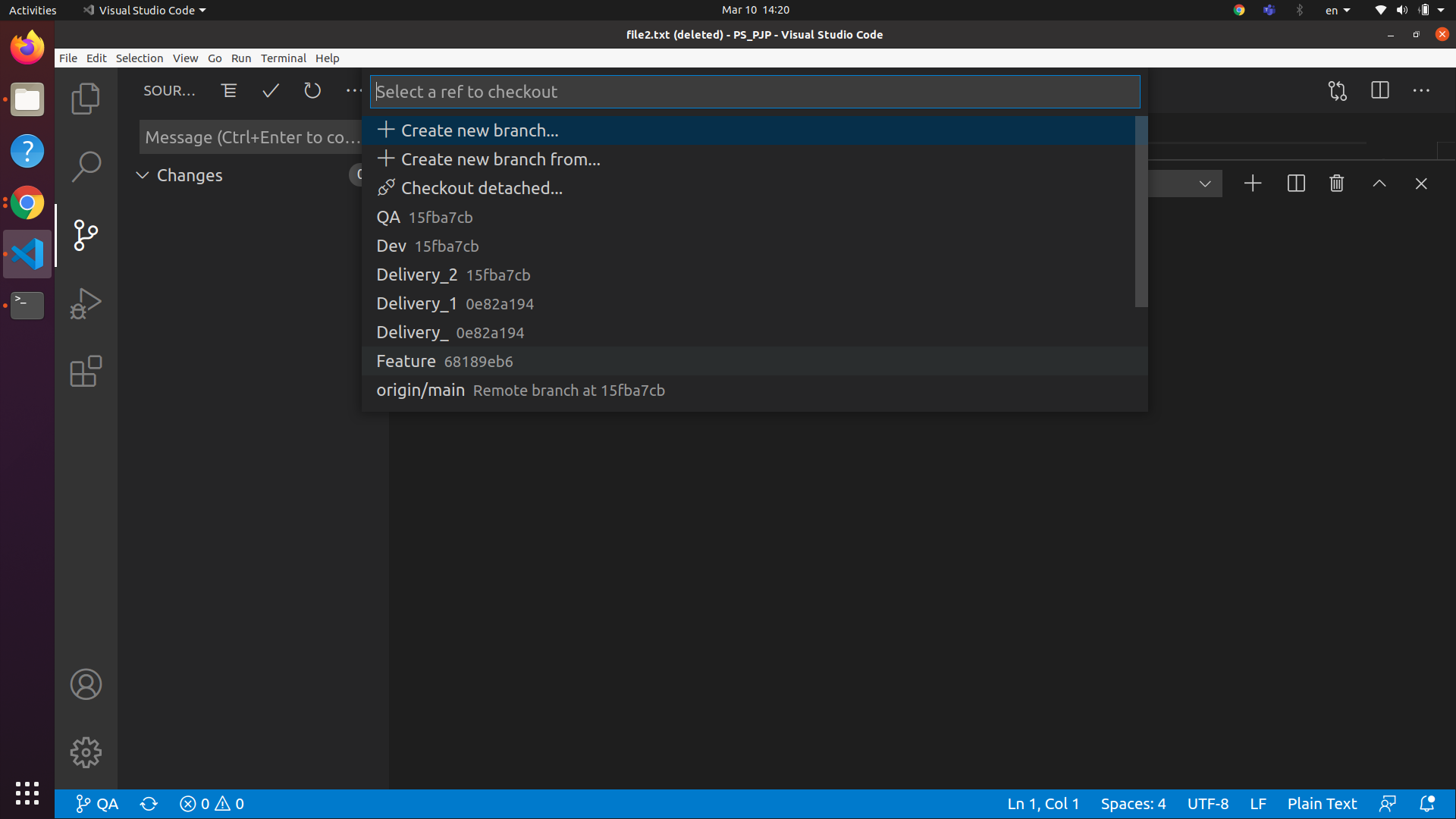
Push the latest code to origin

git push origin “name of branch”

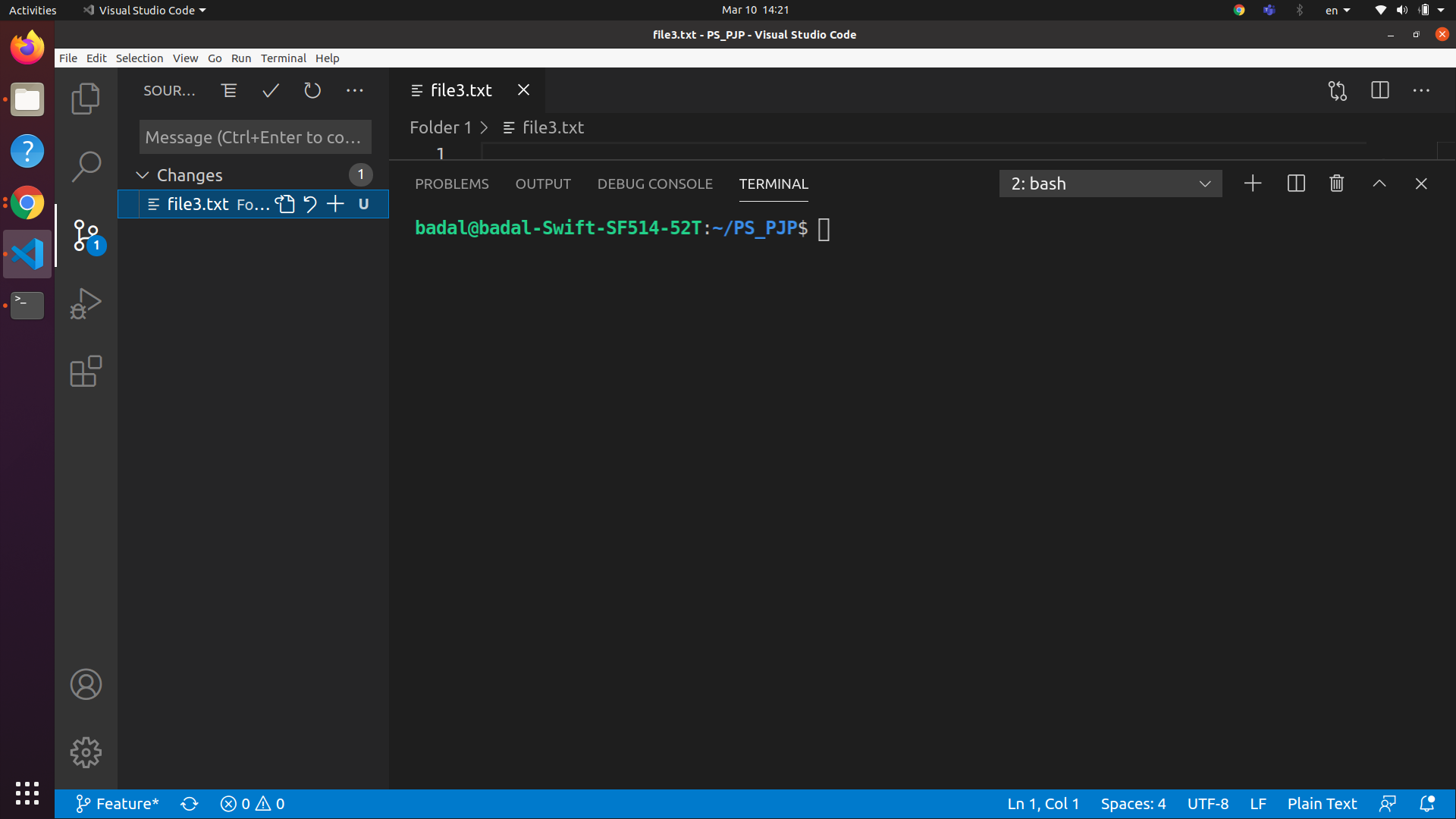
* b

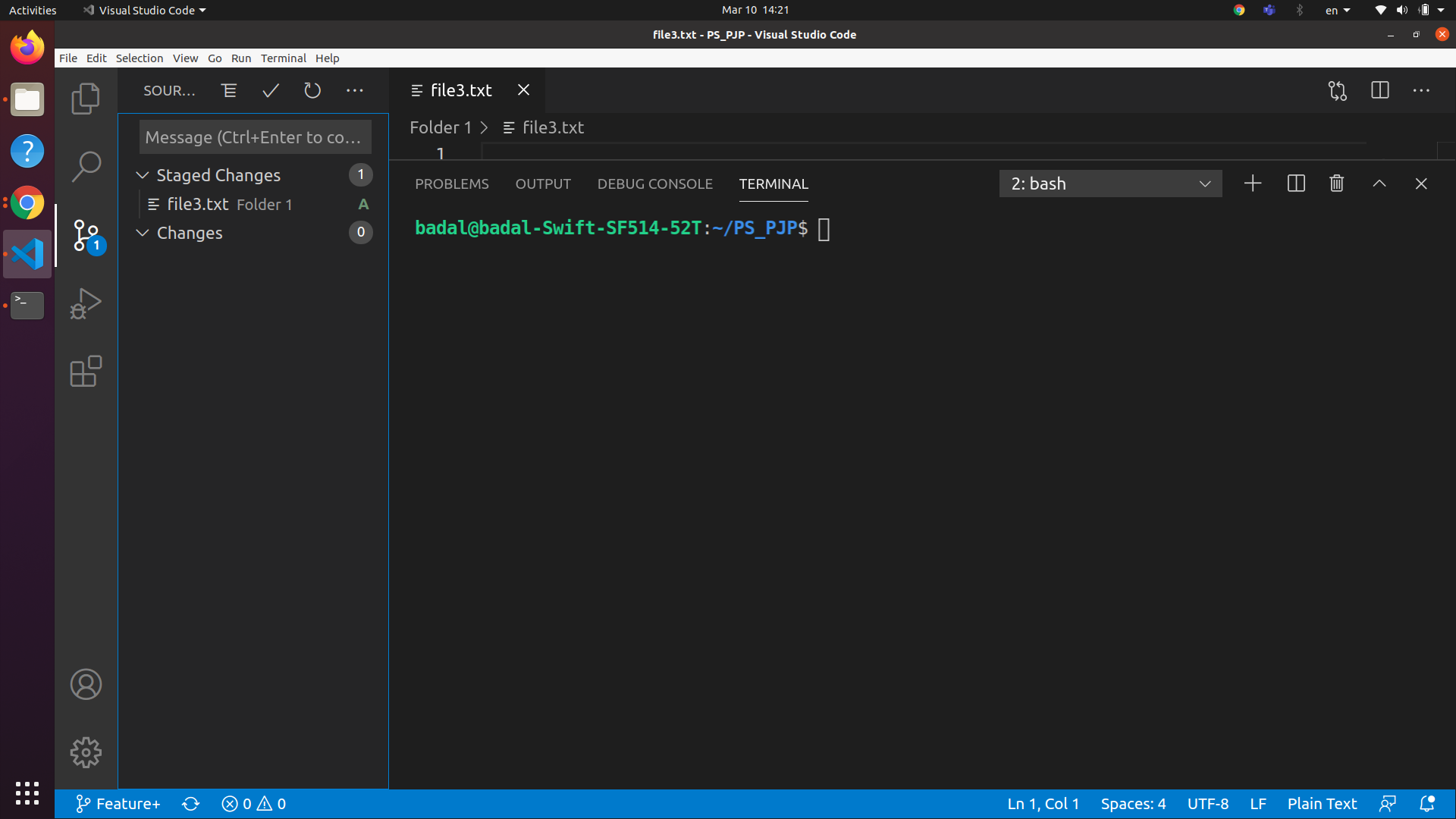
**Perform all the tasks using vs code**

(i) Create new branch

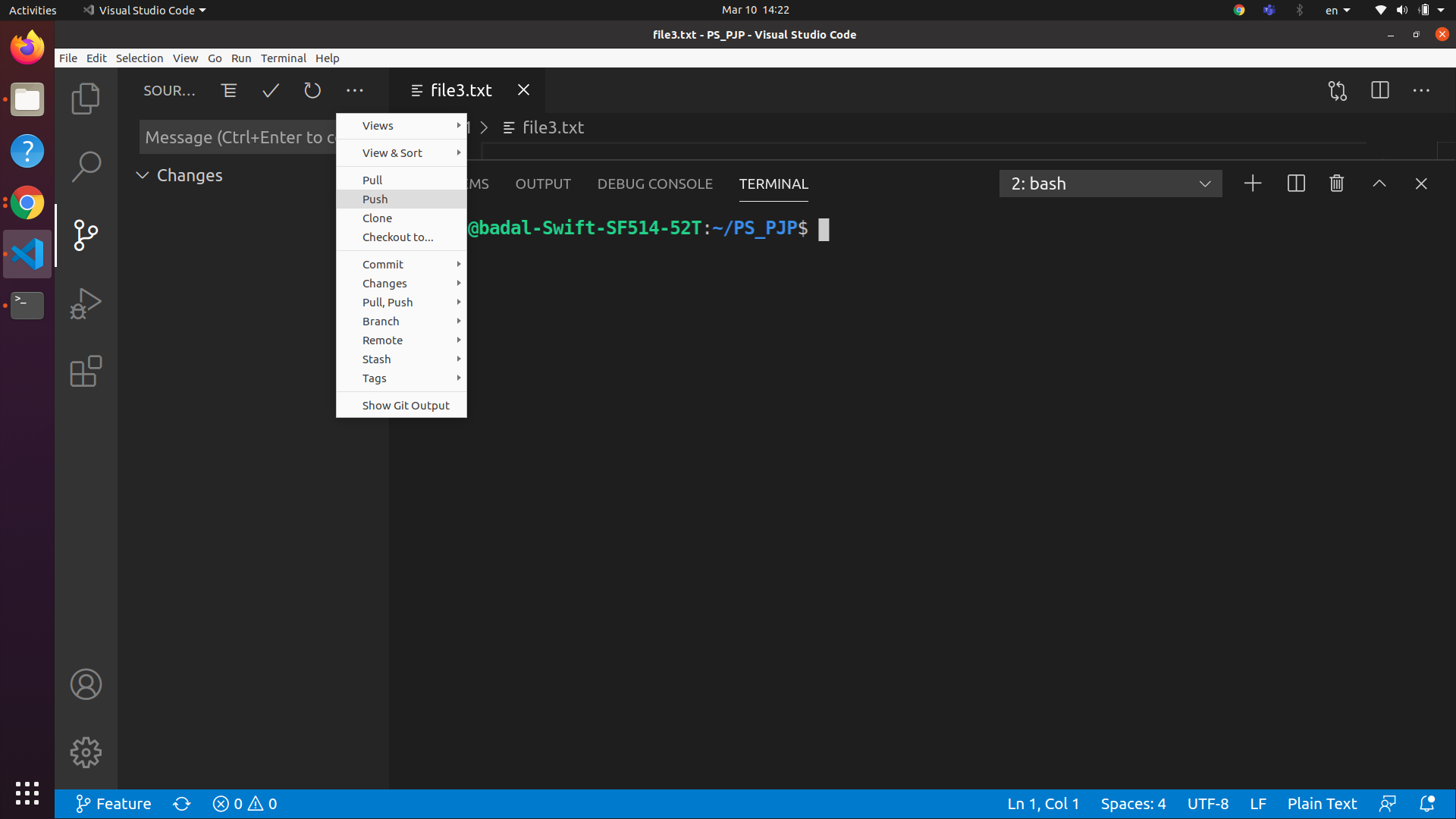
****

(ii) Make changes , add files

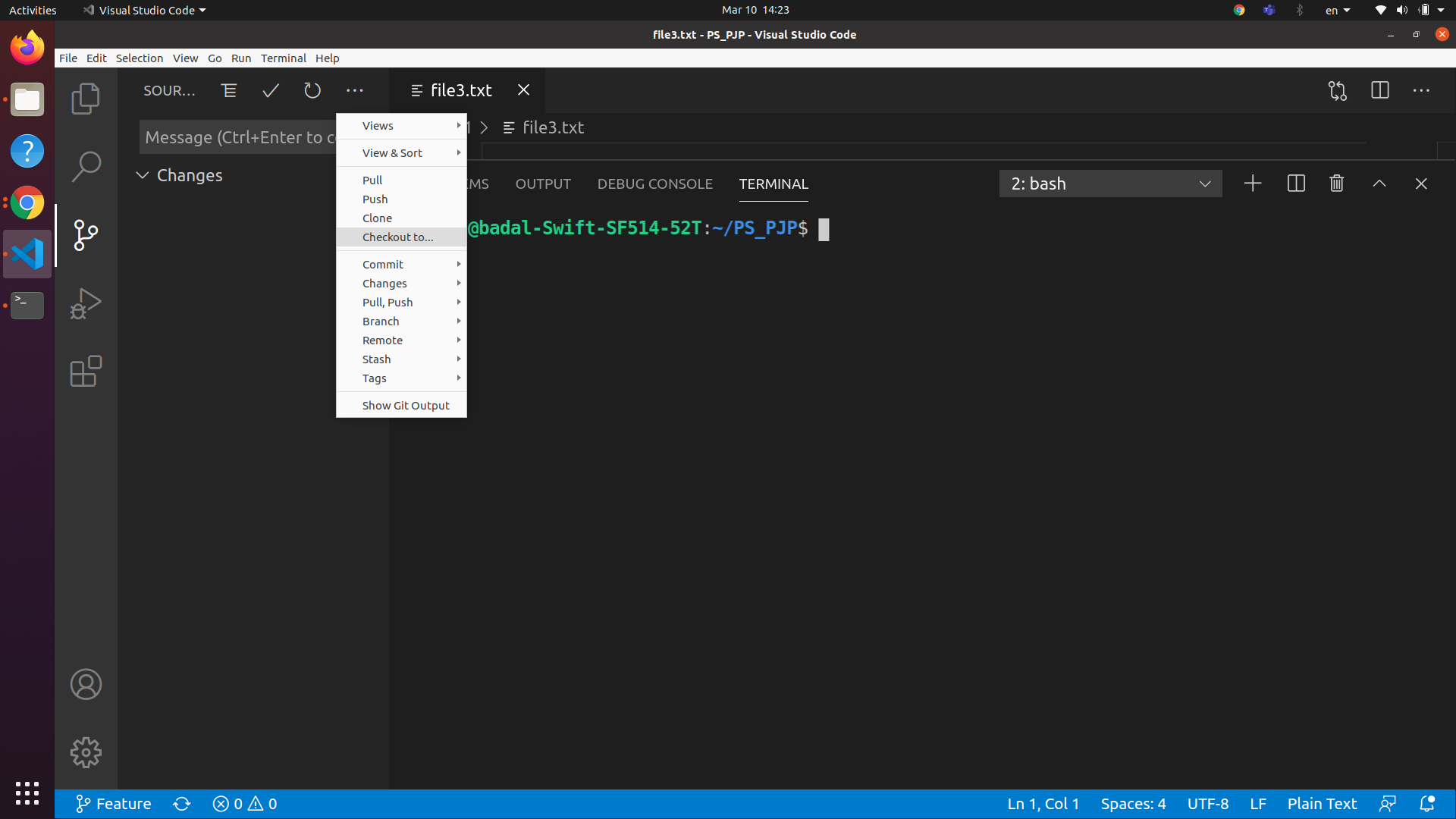


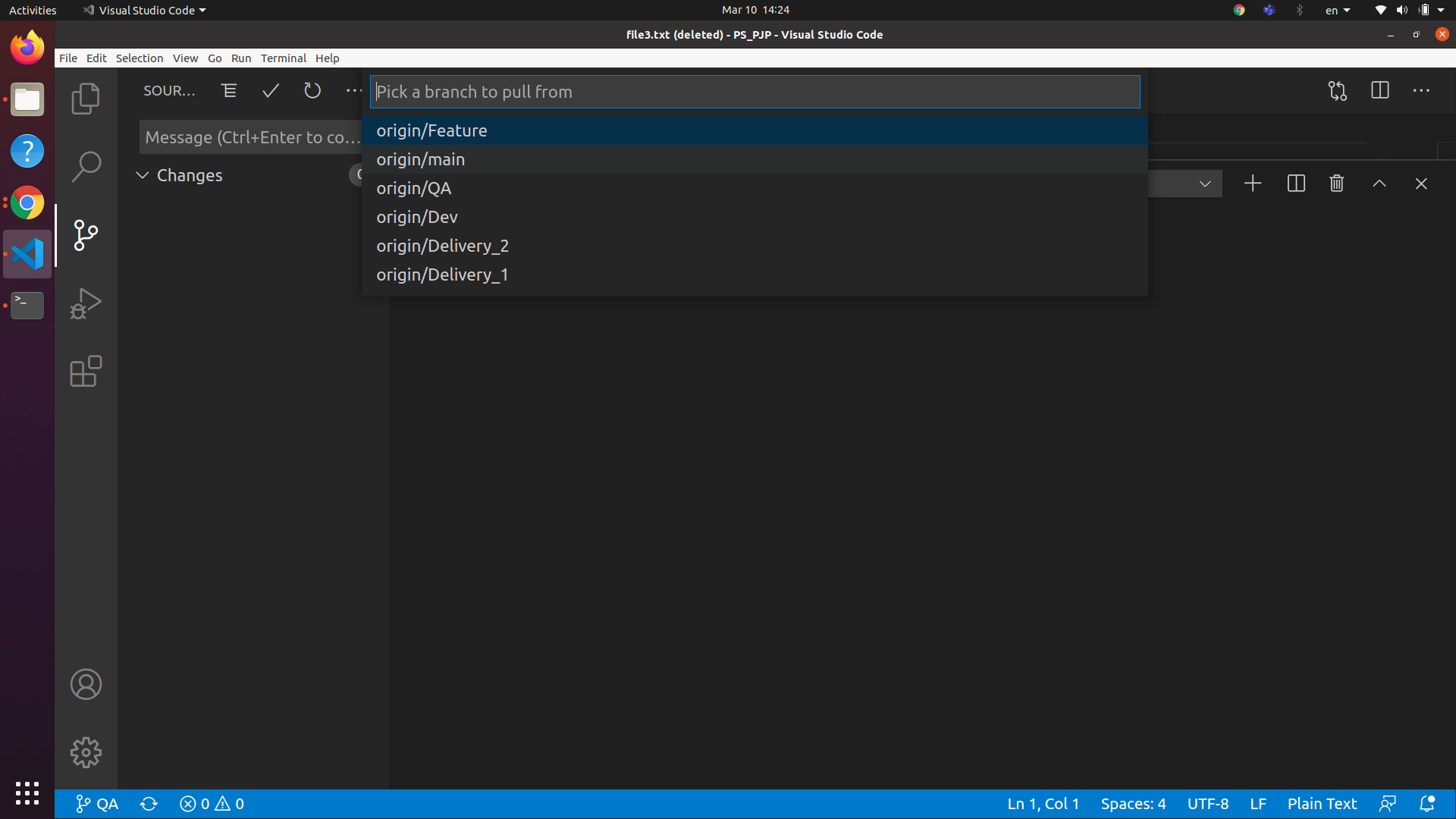
(iii) Stage the changes

(iv)Push the changes

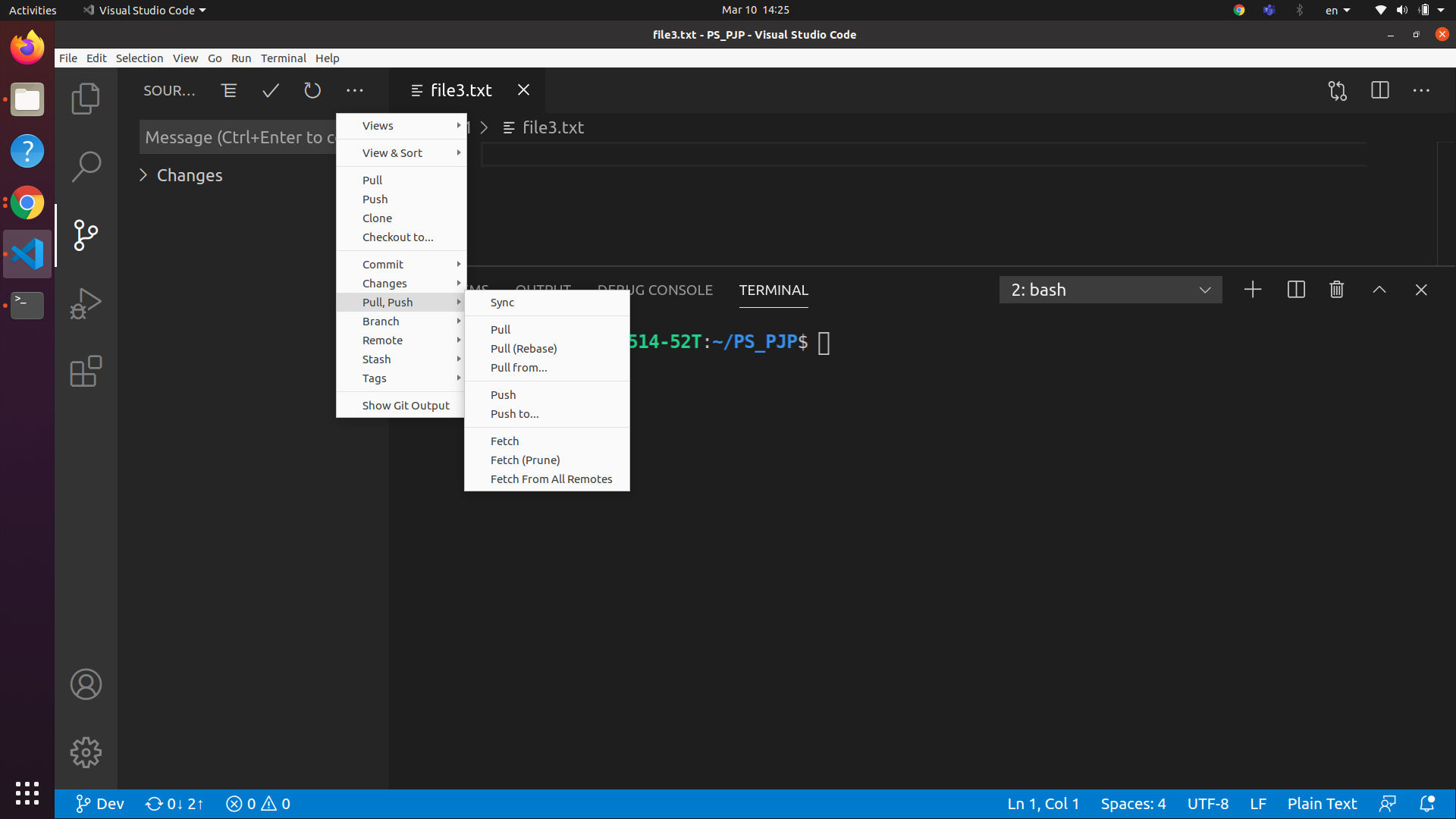


(v)checkout to a specific branch





(vi) pull the code from remote



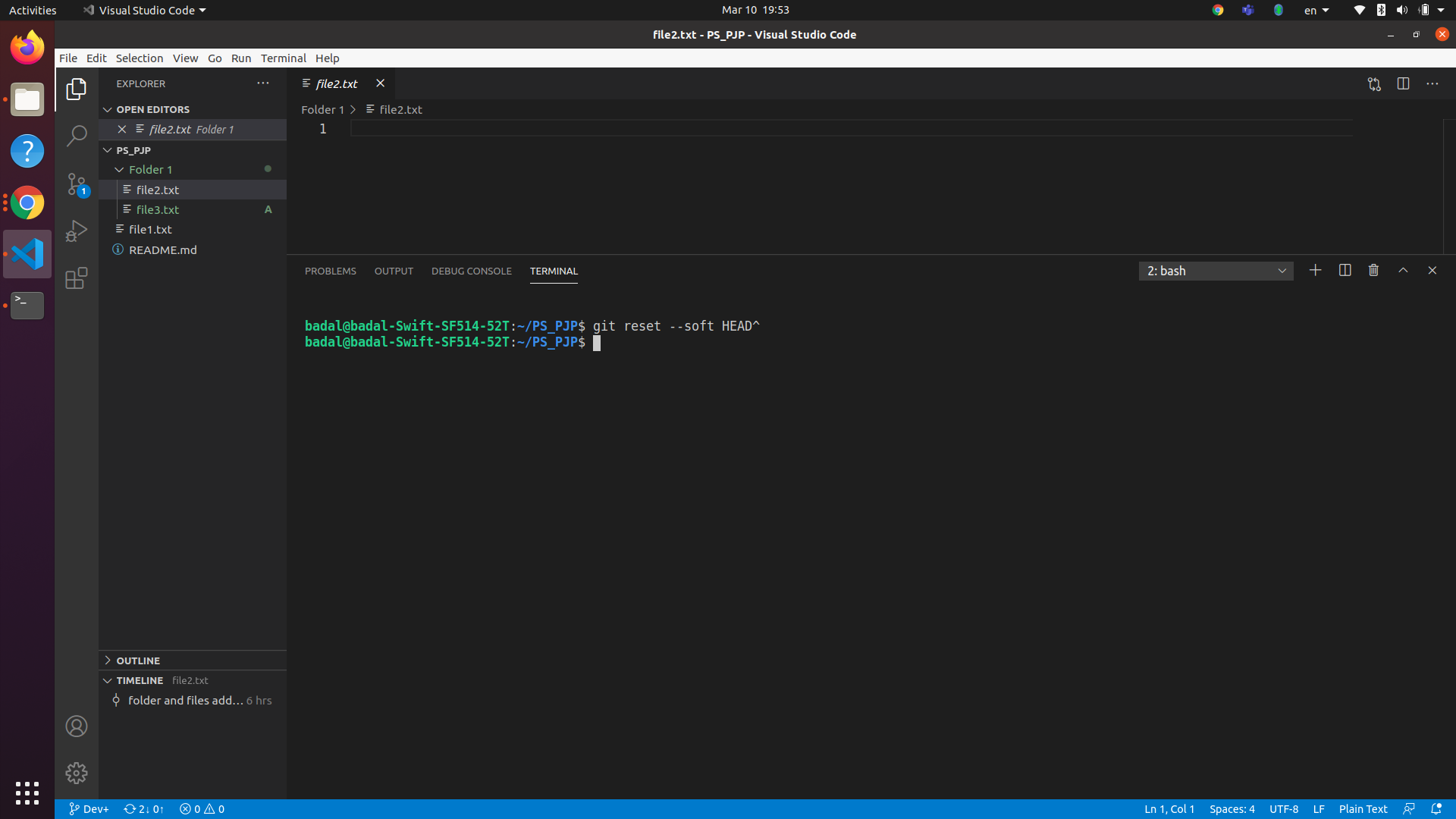
* C

(i) Reset hard and soft

Soft reset : -

git reset --soft HEAD^ (soft reset the last change)

Reset the head to specific state with putting the changes staged



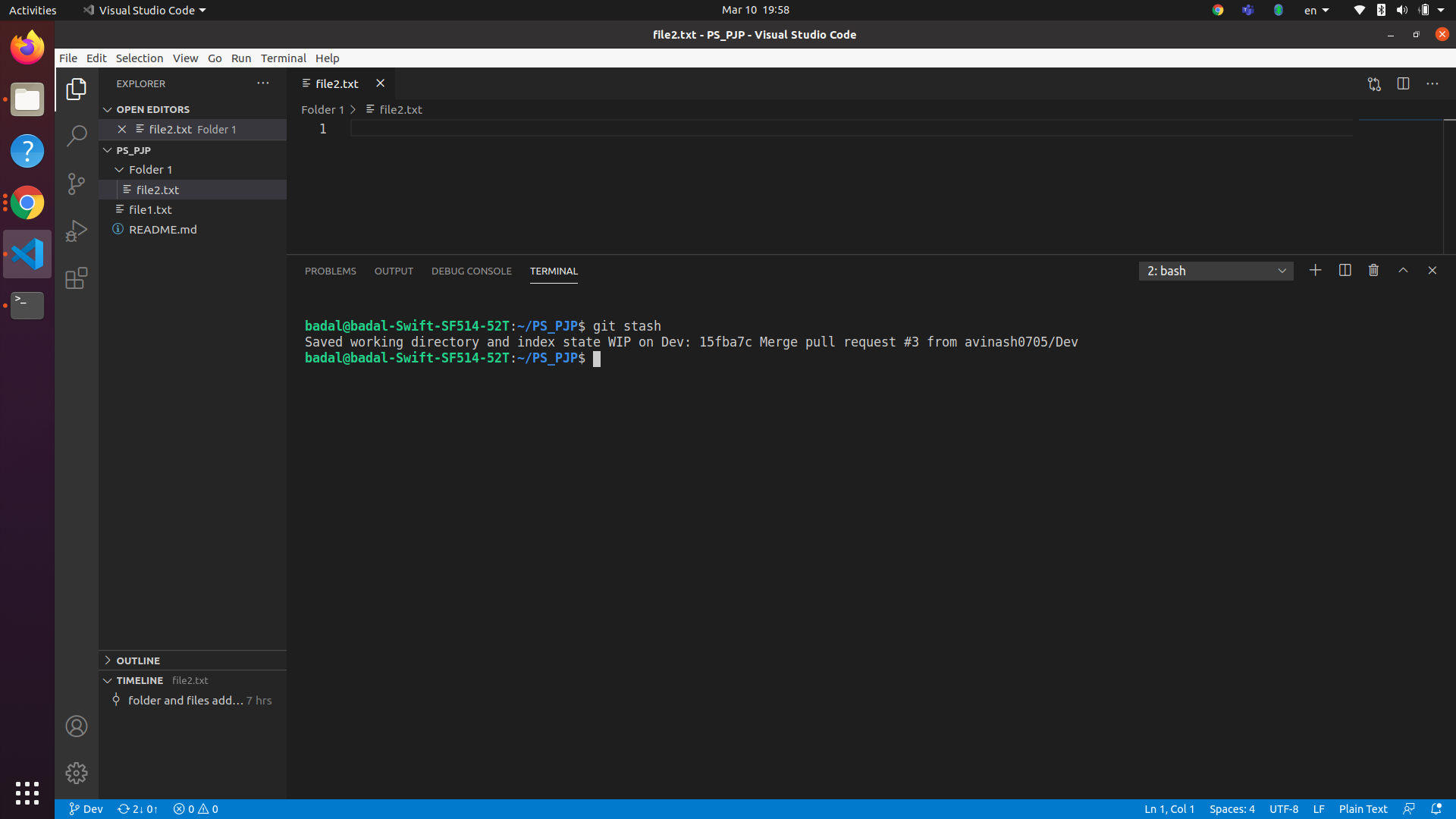
Hard reset :-

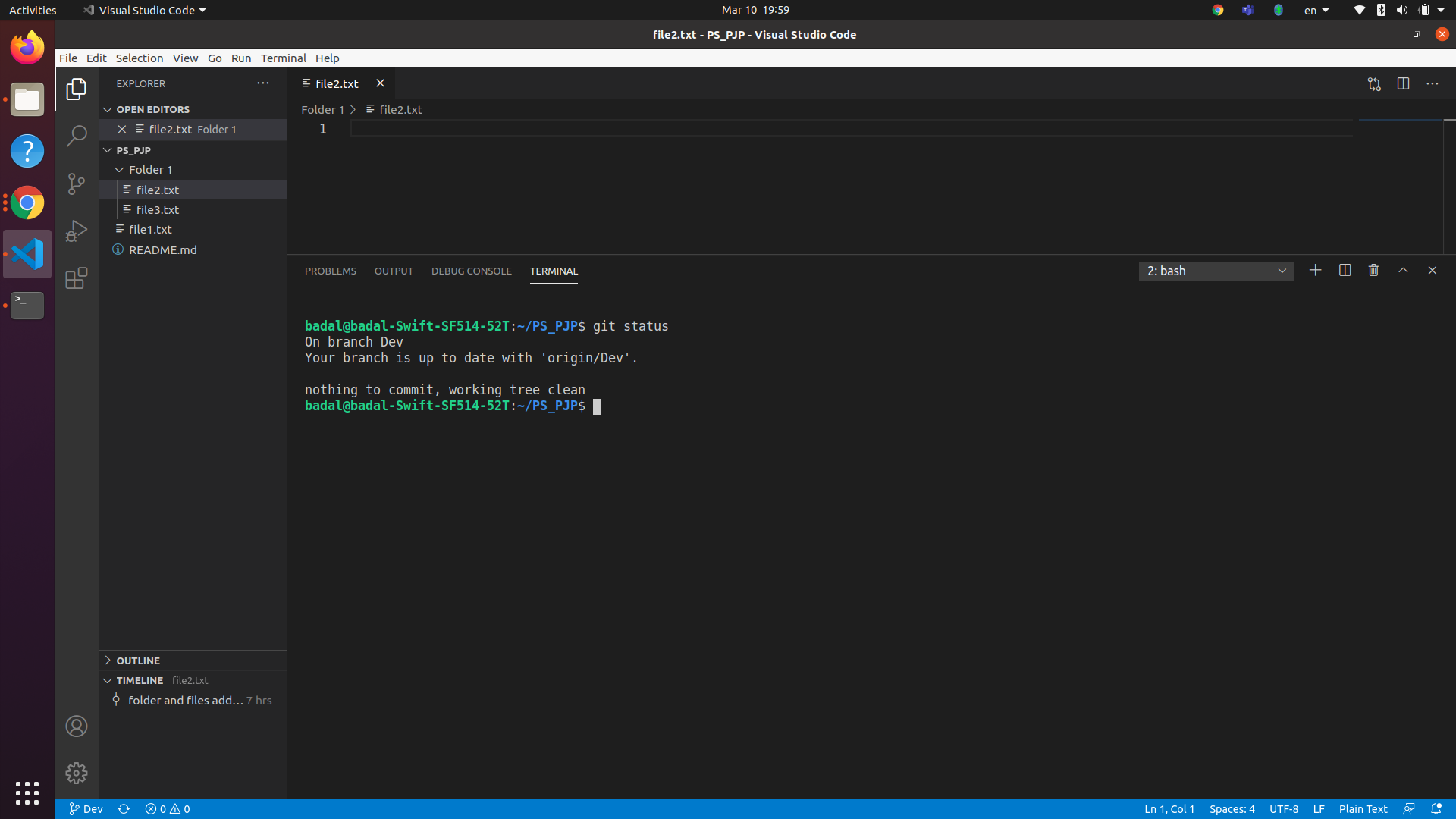
git reset --hard HEAD^ (hard reset the last change)

Reset the head to specific state while getting rid of changed done

(ii) Stash the changes in local

git stash





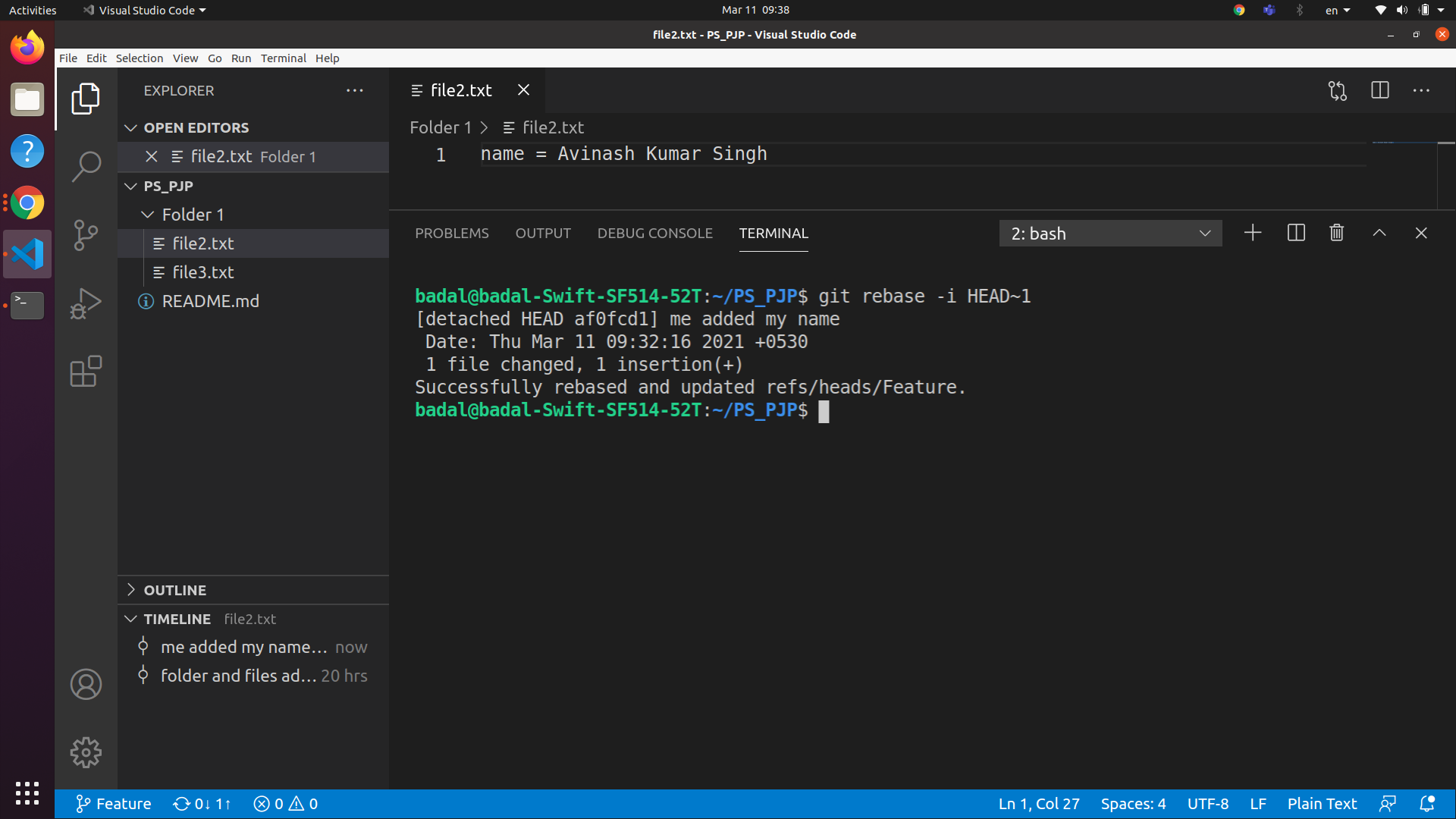
(iii)

**Rebasing:**

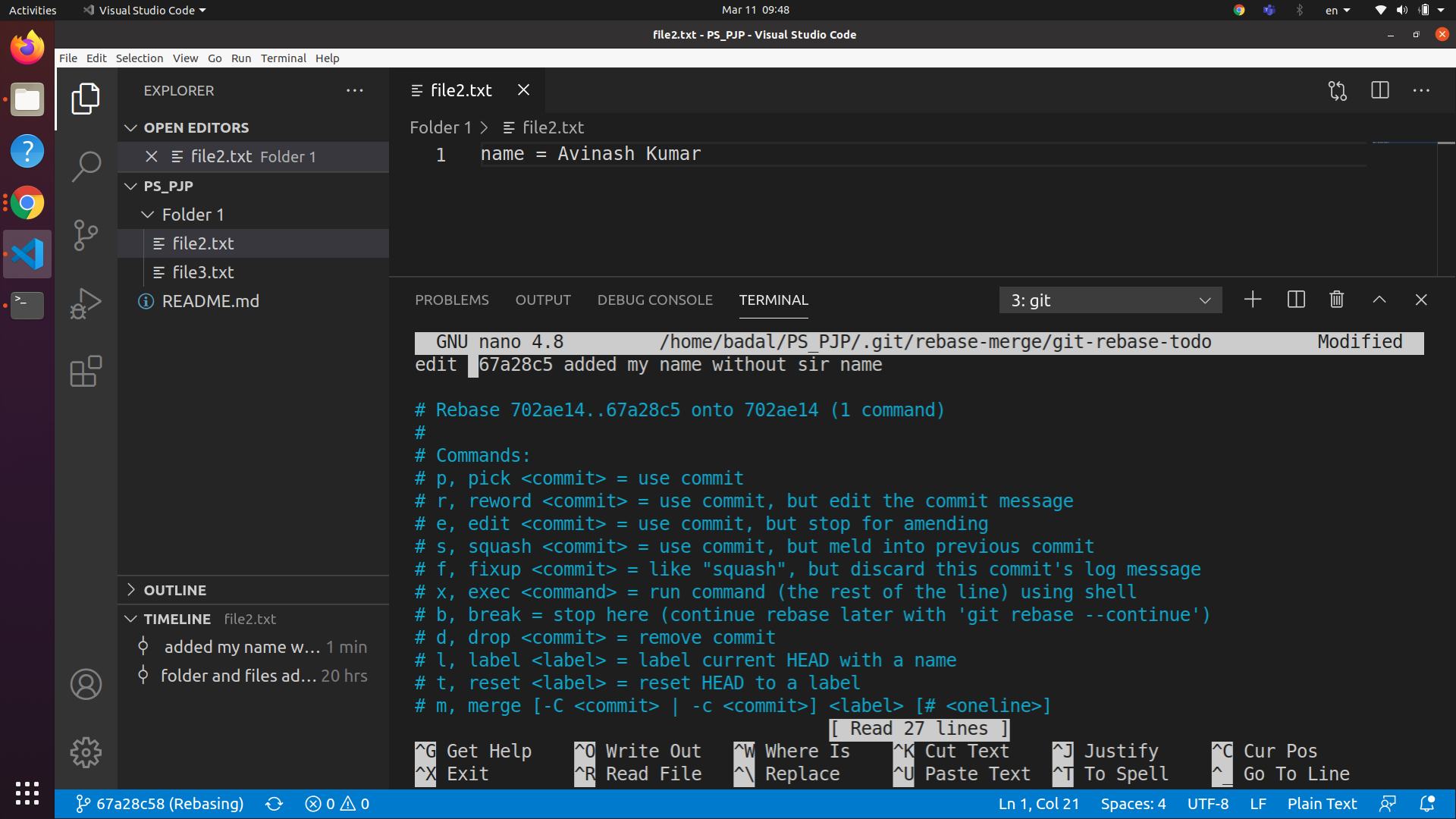
**Cmd :-** git rebase -i

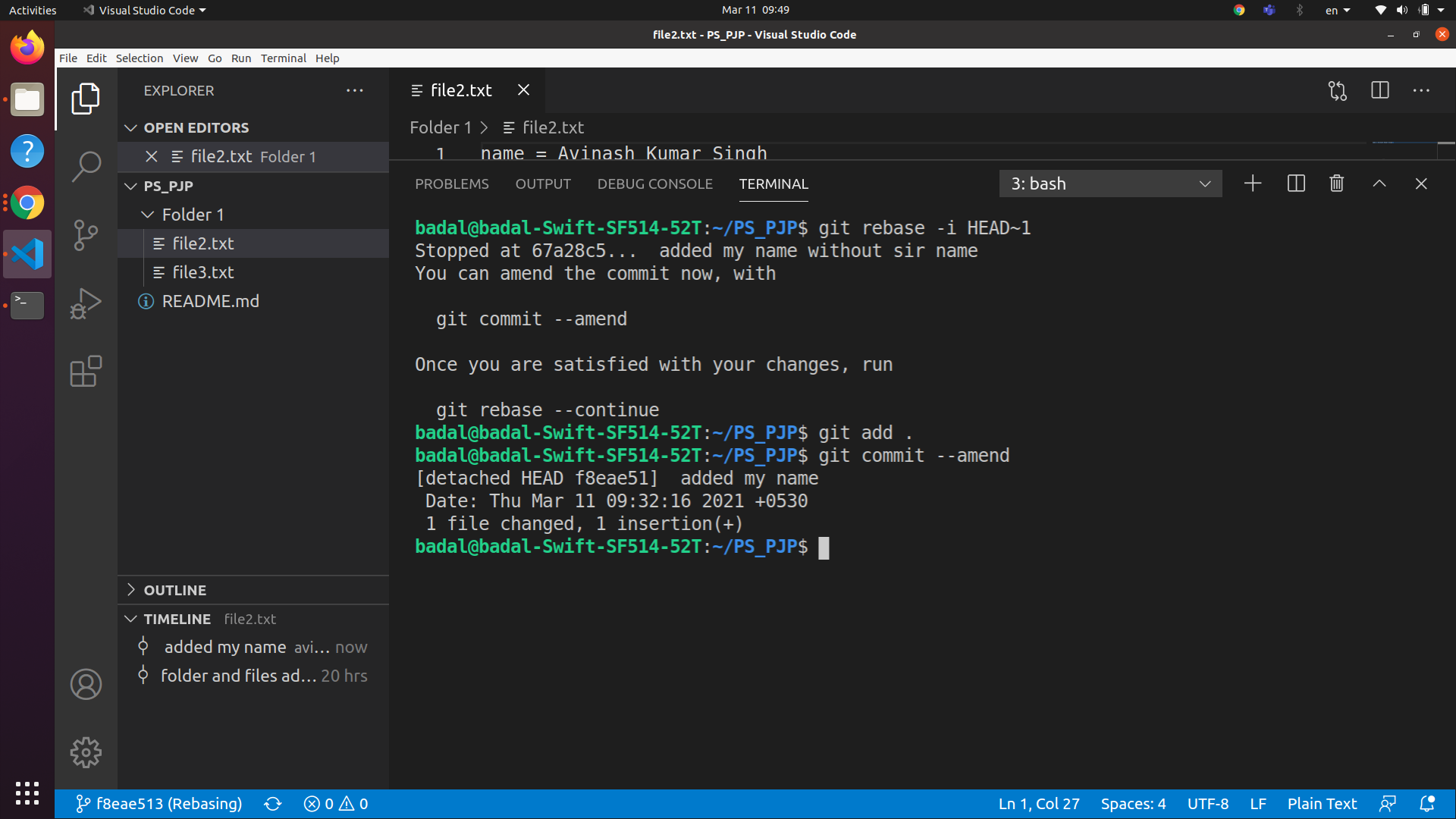
To get all interactive rebasing

1. **Reword:-** it provide a way to alter the commit message while rebasing

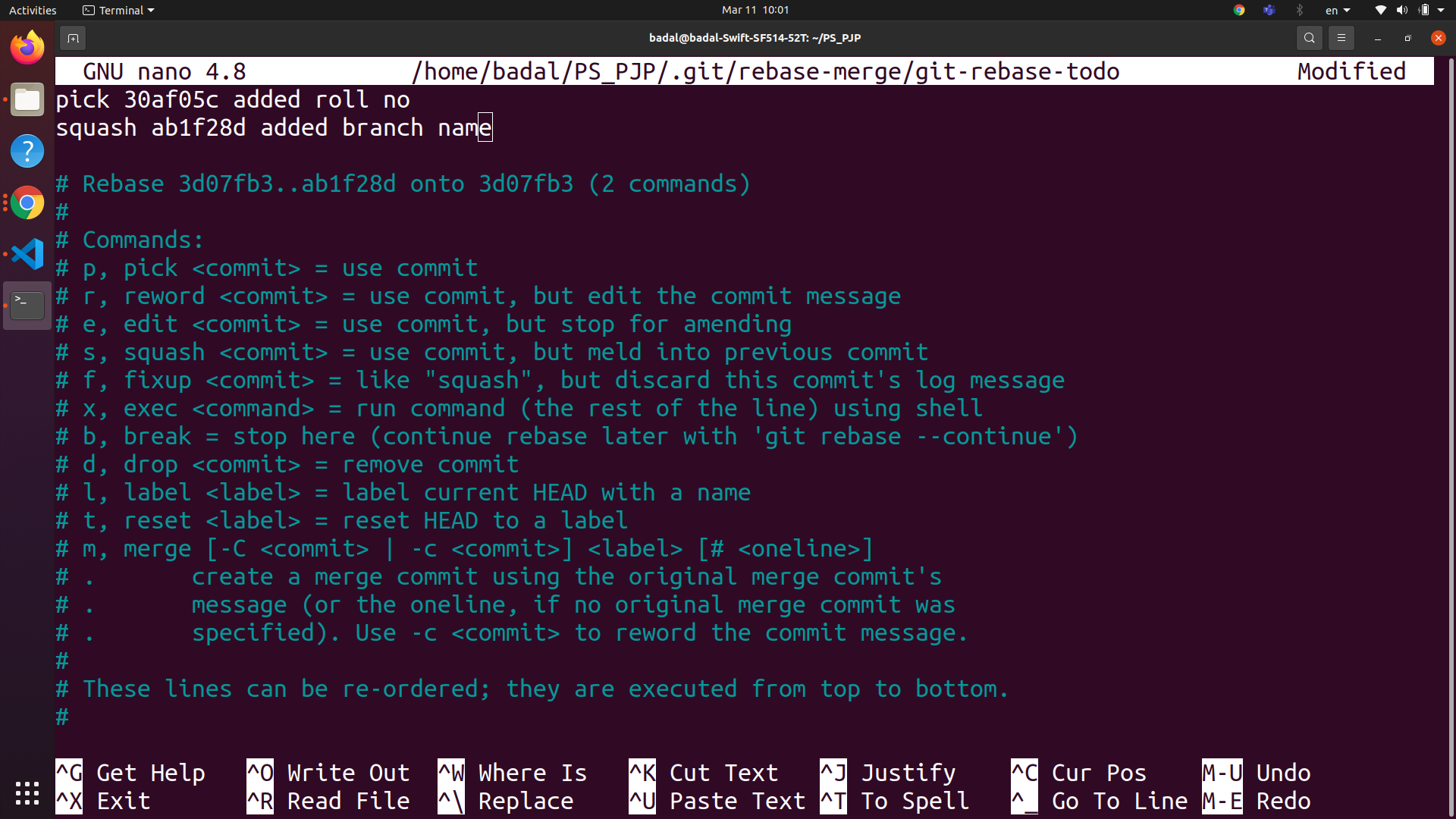


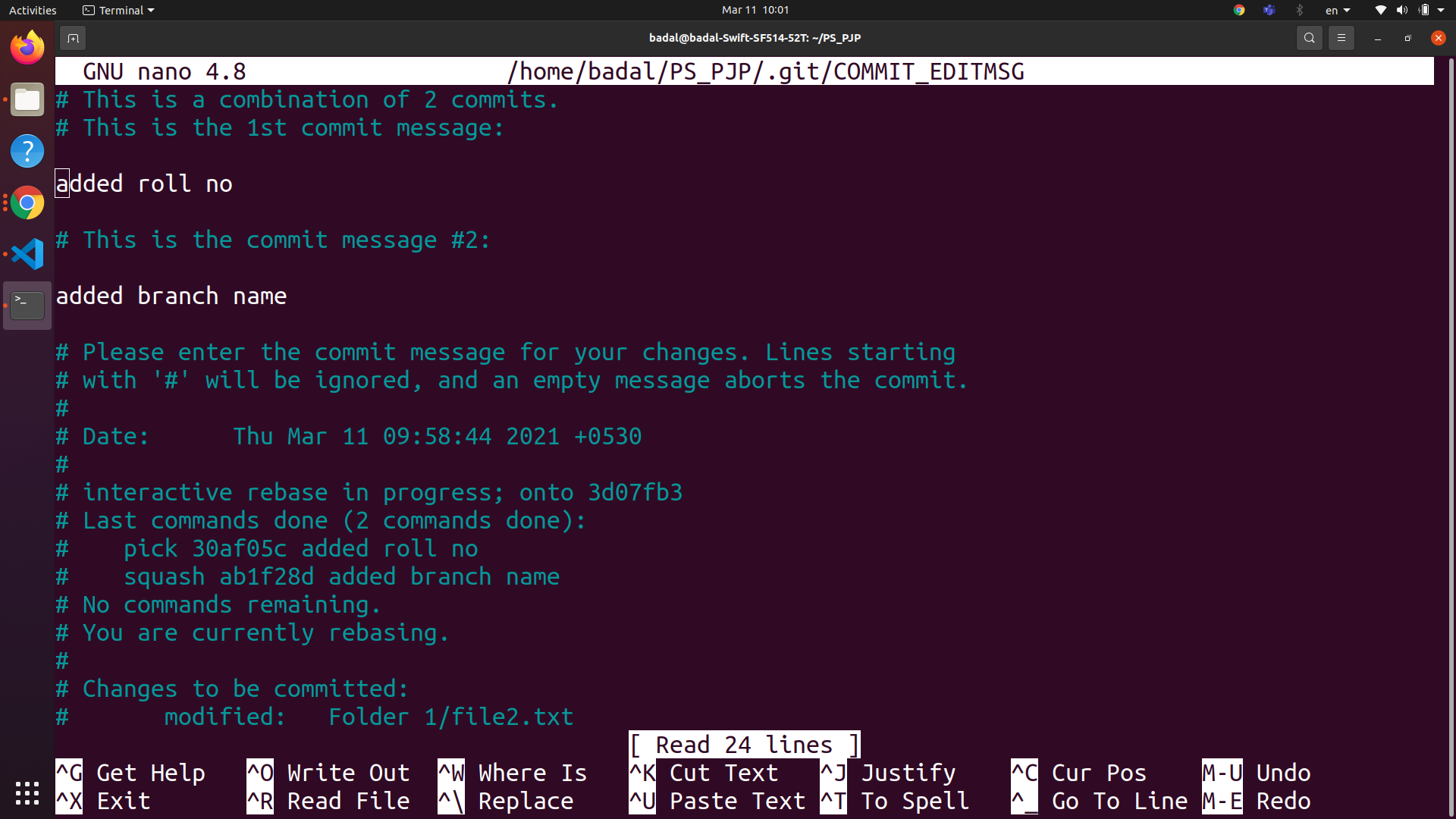
1. **Edit :-**  it facilitates us to change some erroneous commit or split a large commit into a single one or we can also make additional commit before rebasing

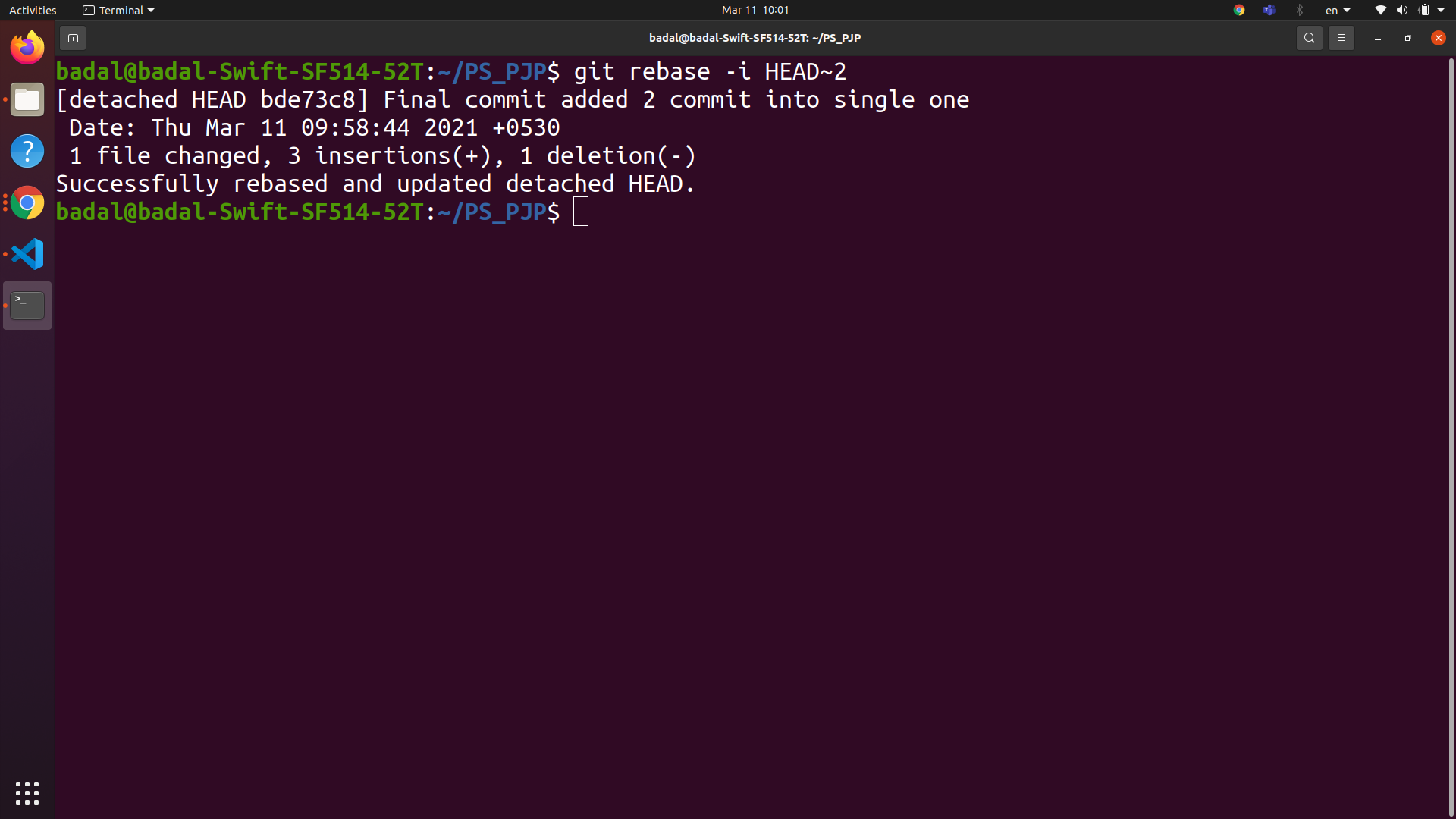




1. **Squash :-** It allows us to combine two or more commits and also rewrite a single commit message for all

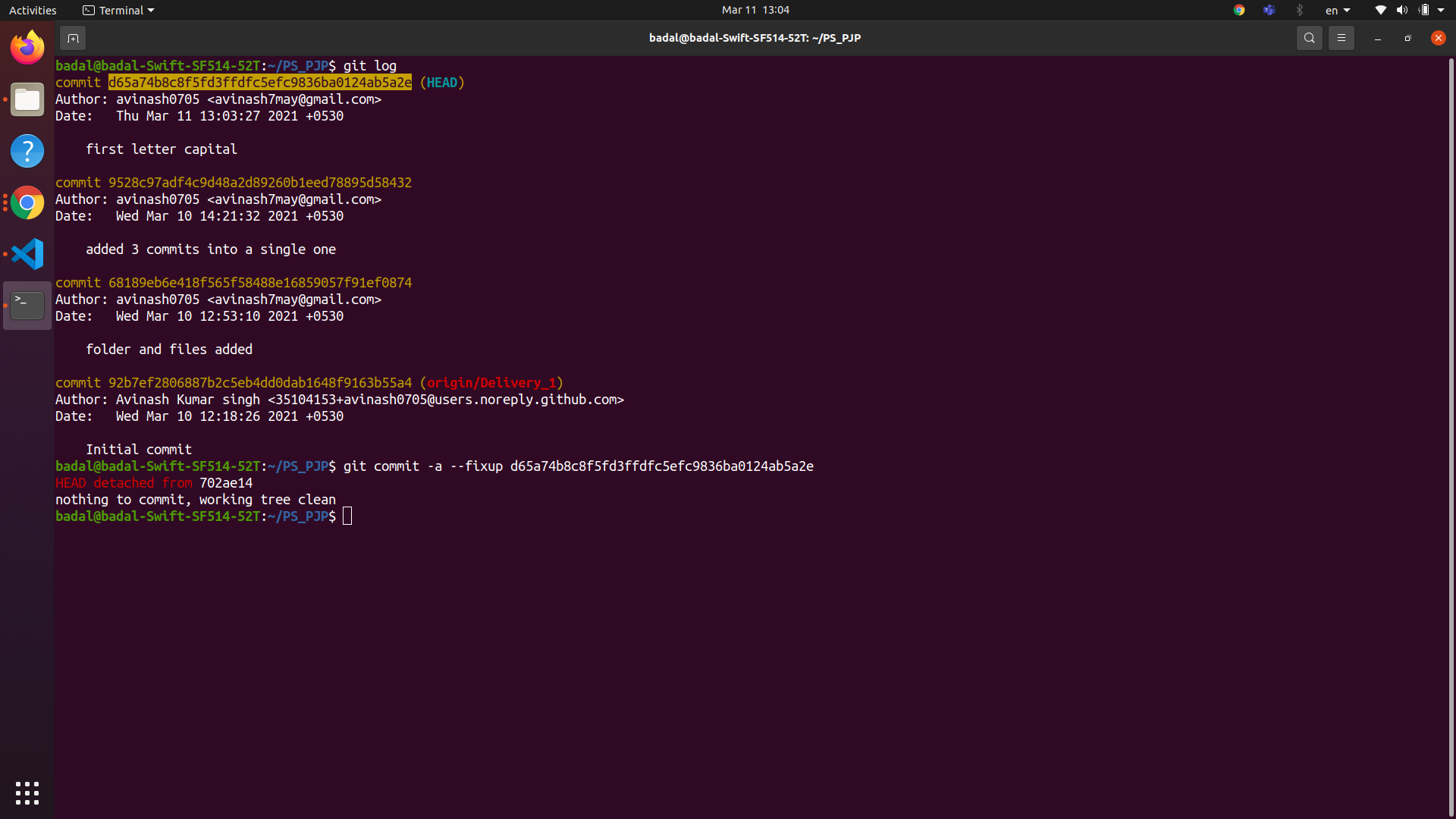






1. **Fixup**:- it discard the message of commits to be merged and the older commits message is used to describe both the changes

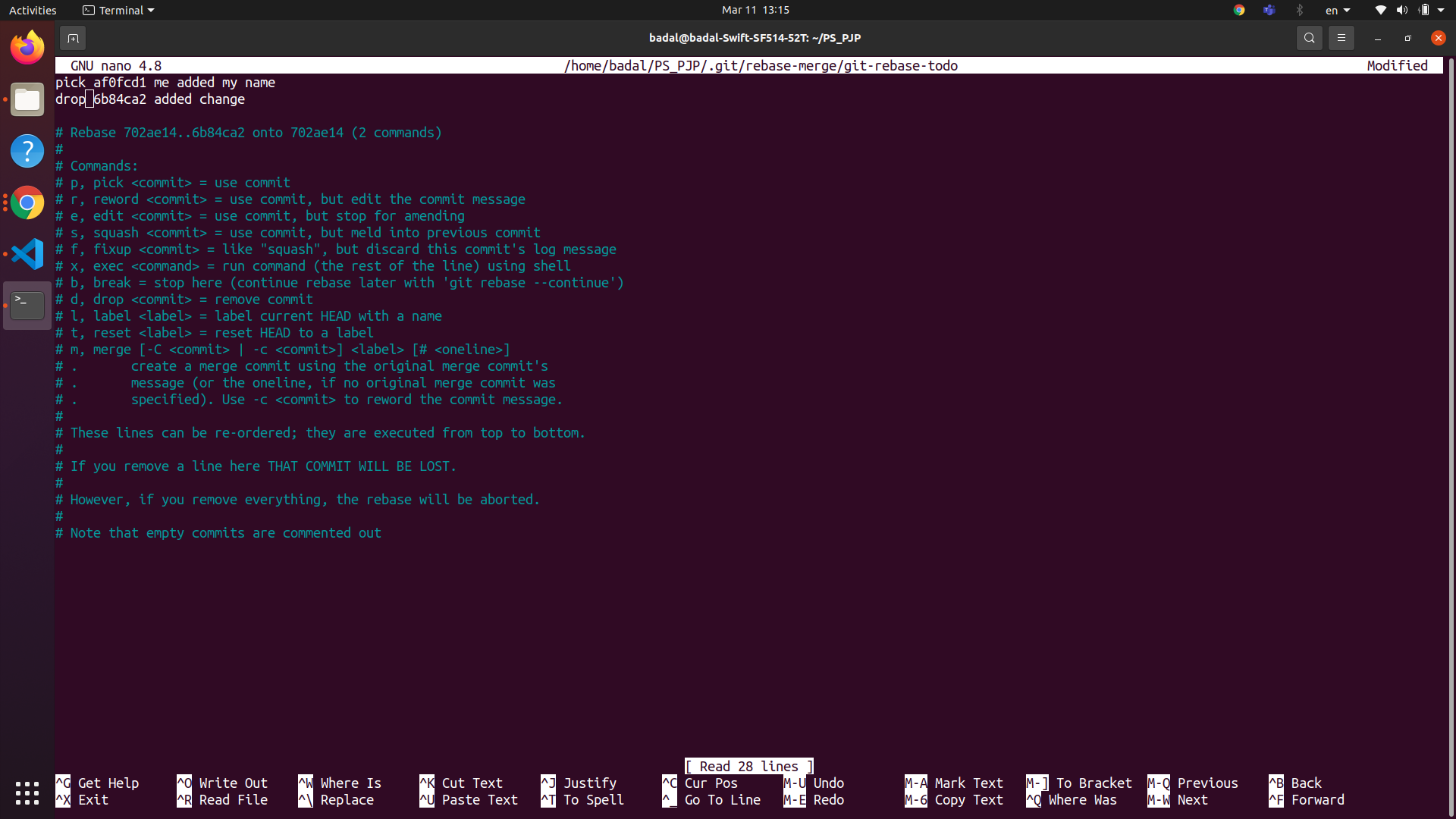
General cmd :- git commit -a fixup “commit ref”



1. **Exec :-** It allows us to run any shell command after each commit or a specified commit

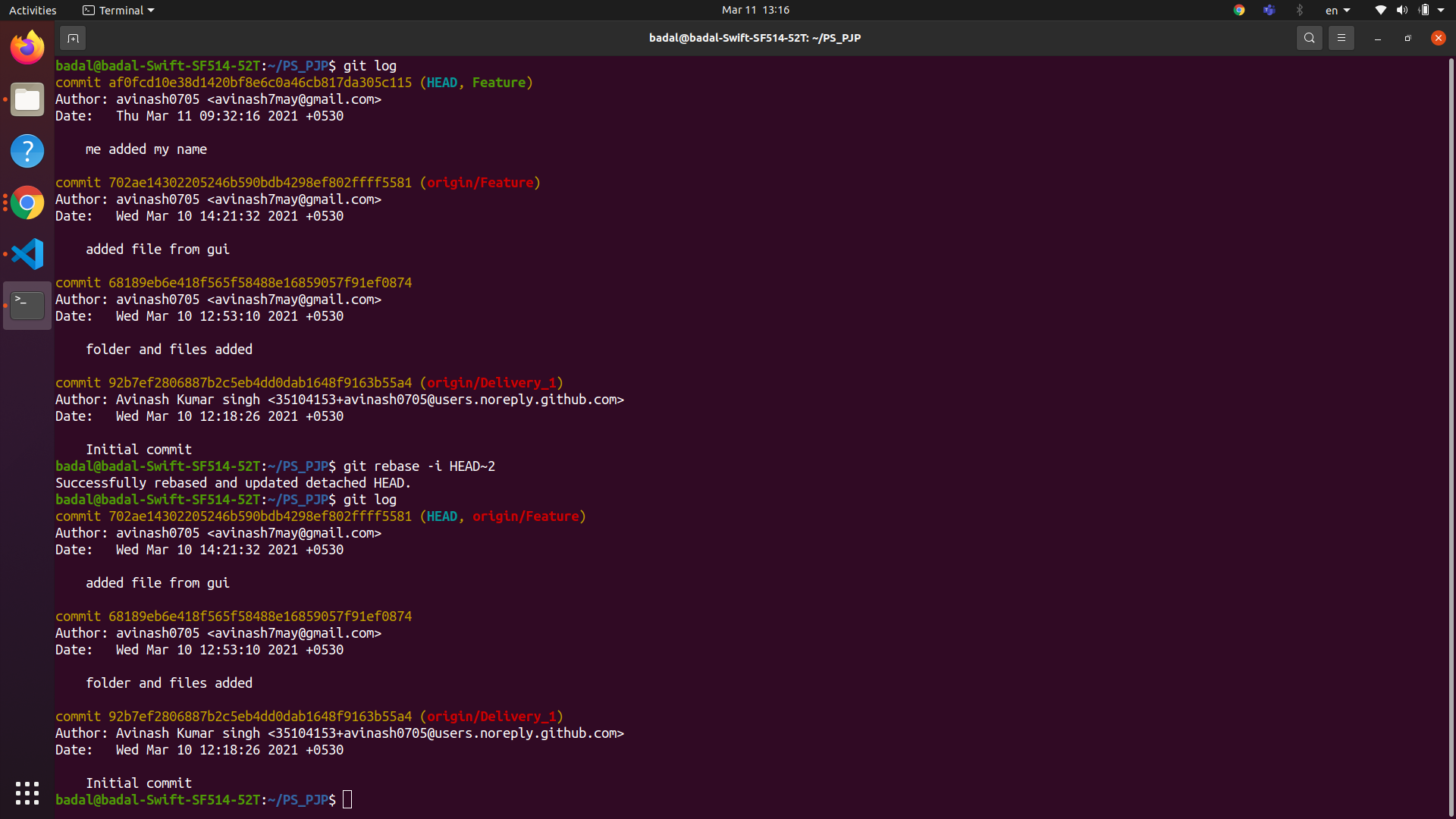
**Basic cmd:-** git rebase --interactive --exec "cmd" some-ref

1. **Drop :-** it is used to remove the commit

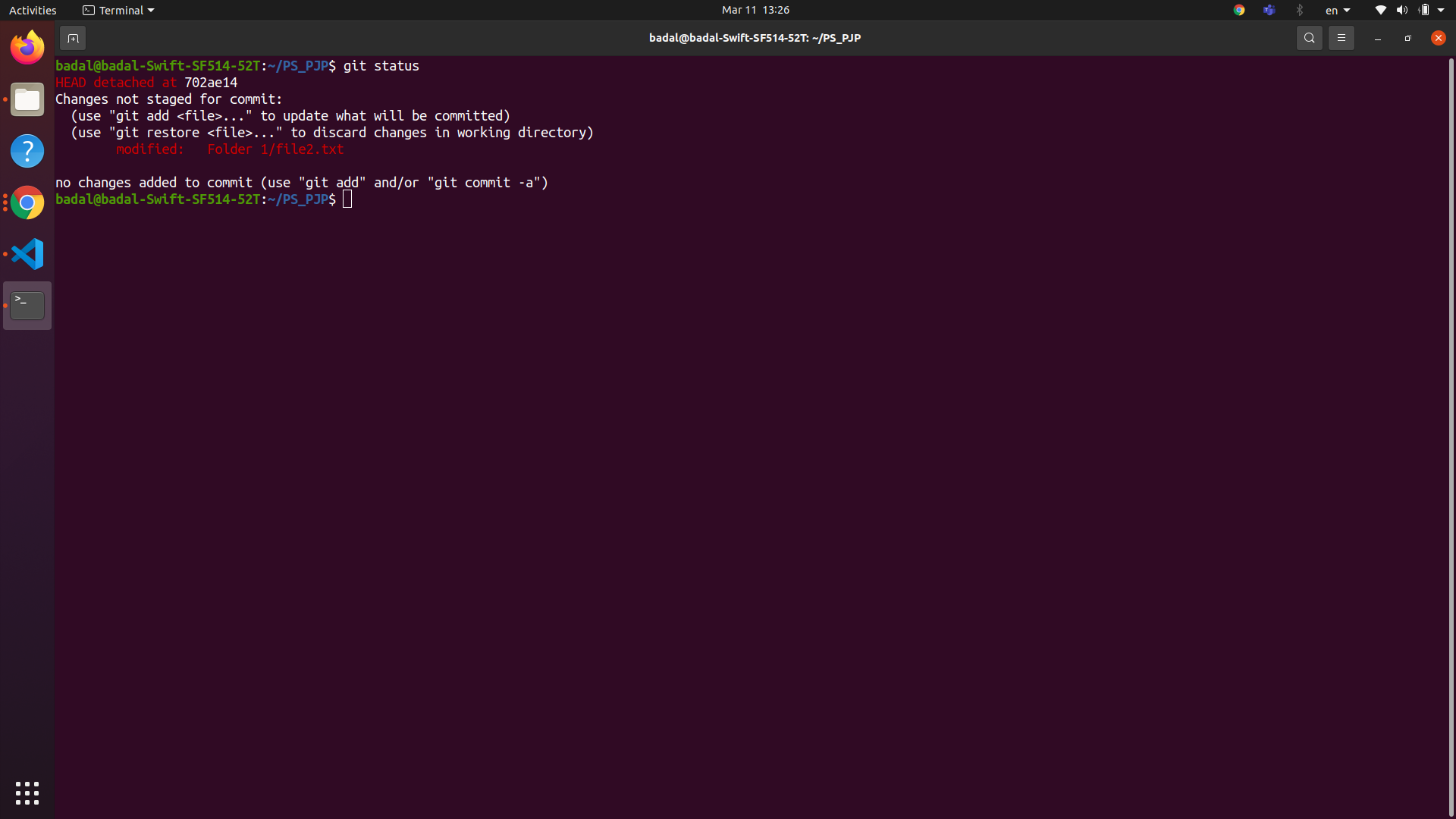
****

**(iv) Git log, Status and reflog:-**

Git log :- git log shows the current head and its ancestors

****

**Git status:-** It displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git

****

**Git reflog :-** The reflog is an ordered list of the commits that HEAD has pointed to: it's undo history for your repo.

