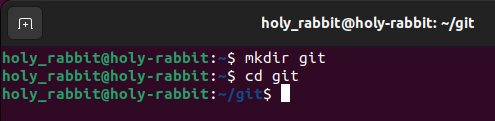
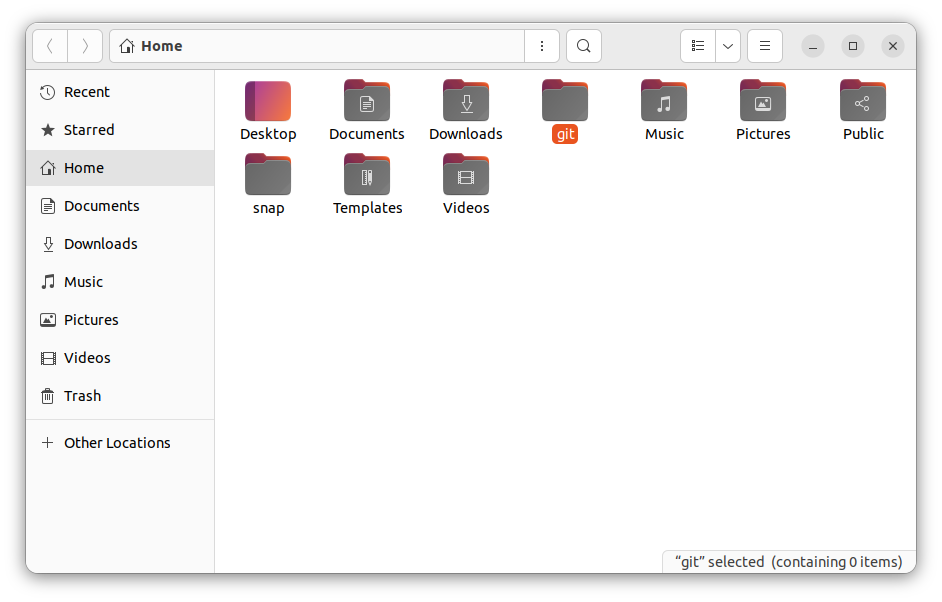
**PROJECT MANAGEMENT WITH GIT**

**1)Setting Up and Basic Commands**

* Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.

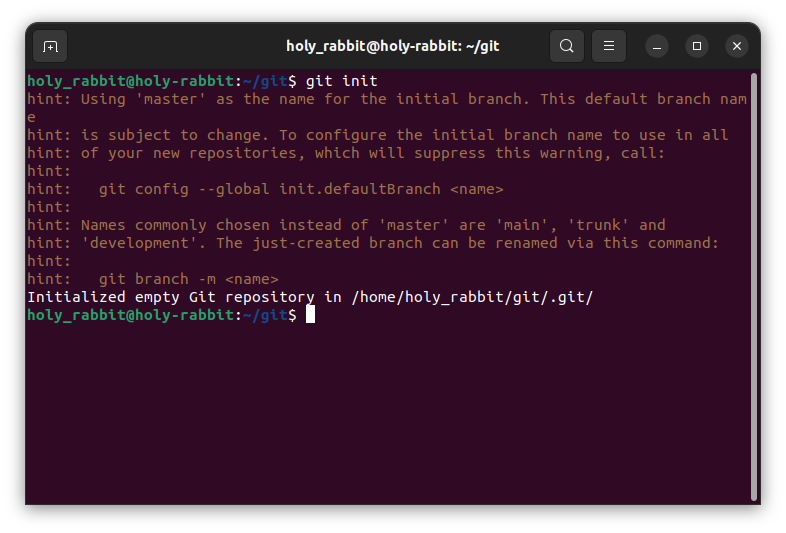
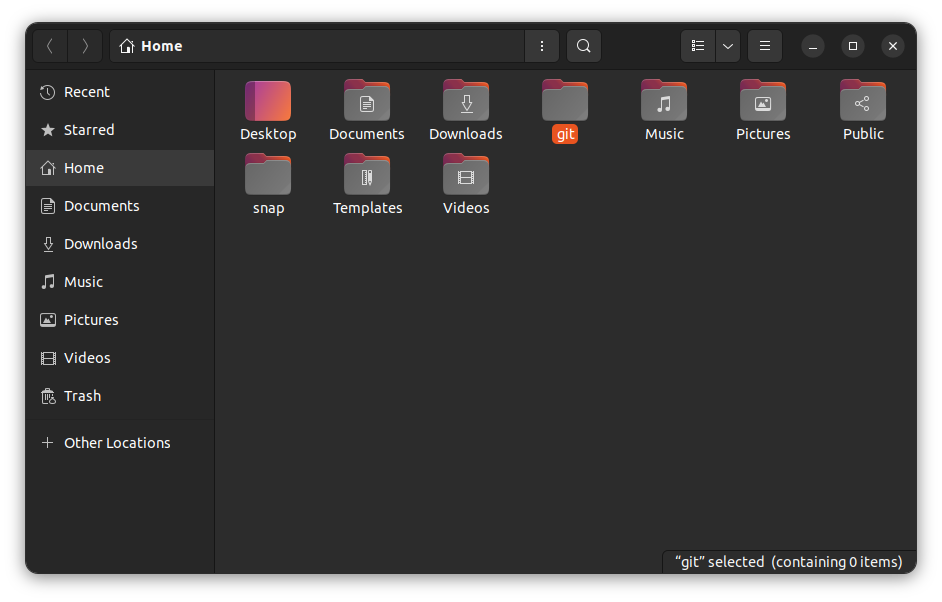
**Step1:**





* cd c:  changes the current working directory into c disk
* mkdir git  creates a folder/directory in the present working directory.
* cd git  changes the directory to the git floder which was created.

**Step2:**

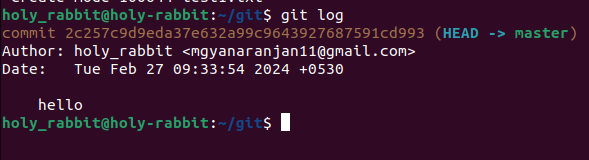


* git init  initializes a empty git repository.
* We can see the .git folder created in the git folder ,in some cases the file is hidden and to see that hidden file we need to click on view the hidden files.

**Step3:**

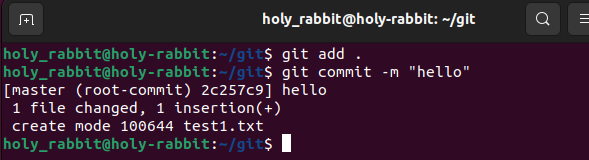
* touch text.txt creates a empty file of txt extension in the current directory.
* D:\git_screenshots\git_01\git_03.pnggit add . /git add text.txt  stages the file in the case of specific file add ,or add . will stage the whole files in the current directory and are ready to be commited.

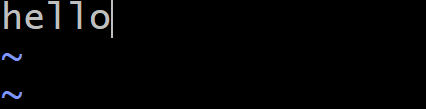
**Step4:**



* git log  displays all the history of commits with commit messages along with the author name and email.
* Every commits has a unique commit ID.

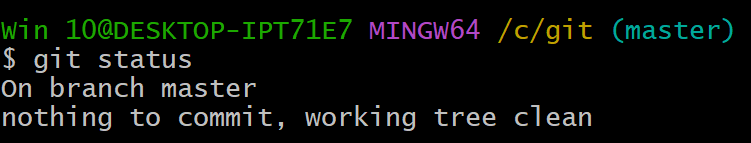
**Step5:**





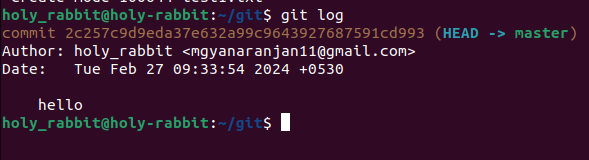
* Here the file content is added using vi text.txt and then file is staged and commited with appropriate message.

**Step6:**



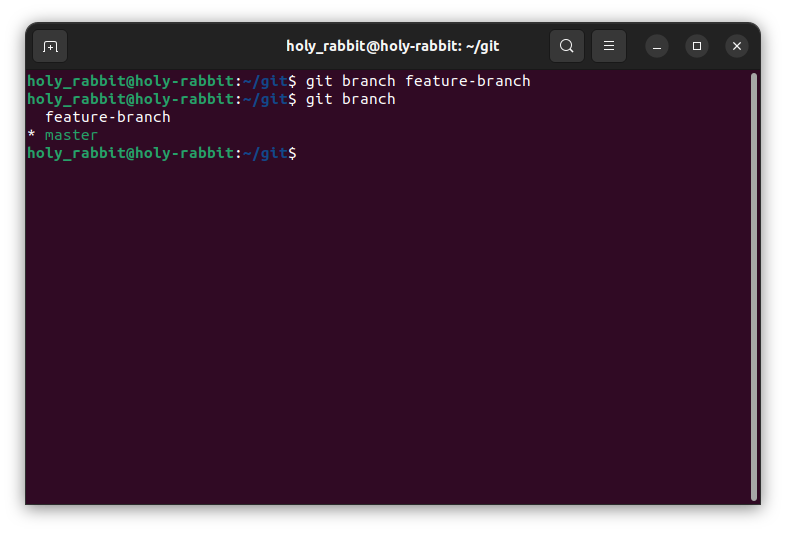
* git status  it checks the status ,like on which branch we are and is there any changes made which are not commited .
* if no any other changes has been made after recent commit the it displays the working tree is clean.

**Step7:**

****

* git log  this will display the history of the commits.

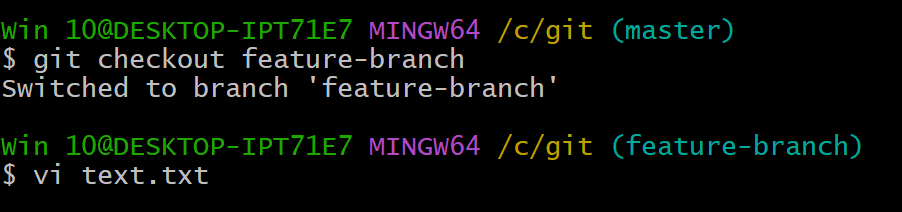
**2)Creating and Managing Branches:**

* Create a new branch named "feature-branch." Switch to the "master" branch. Merge the "feature-branch" into "master.

**Step1:**

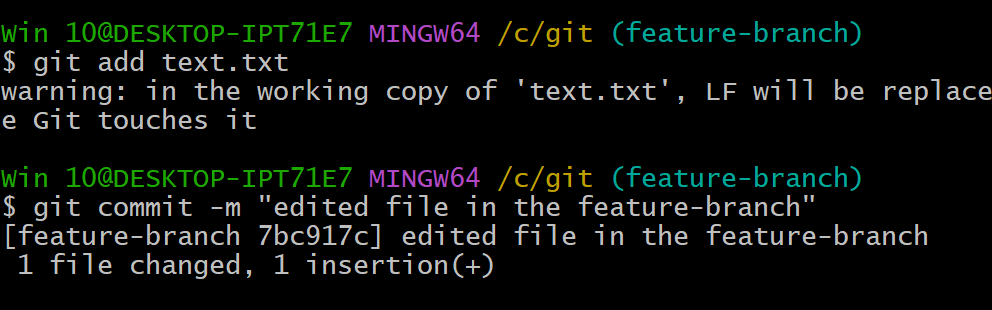
* git branch feature-branch  this will create a new branch of the main master branch in which the contents and files are copied from the master branch .
* git branch  this command will show all the branches we have made and the current branch will be in green colour.

**Step2:**



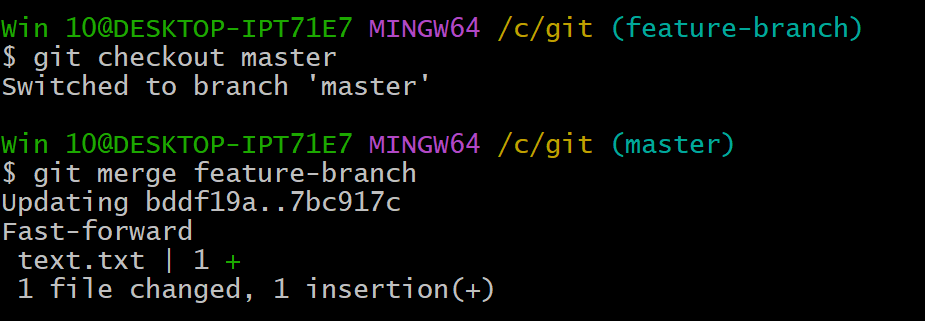
* git checkout  it is used to switch one branch to other branch ,here we are moving from branch master to the feature-branch.
* And also we have edited the file text.txt.

**Step3:**



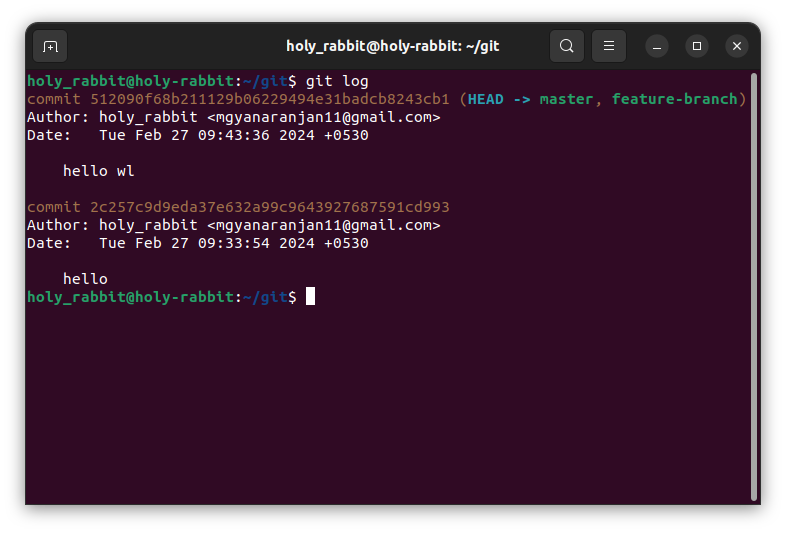
* Here in the feature-branch we staged the file and commited with message saying “edited file in the feature branch”.
* So here the file has been changed ,but in the master branch it will be as it is until we merge the feature-branch with the master branch.

**Step4:**



* For merging the file to the master branch we first need to move to the main/master branch using “git checkout master” command.
* Then we can merge the branch with “git merge feature-branch” command.
* So ,now the files will be meged ,the changes or the edits in the feature-branch will be merged.

**Step5:**

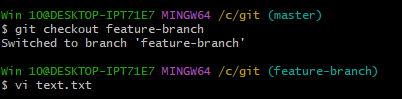


* git log  here the history of the commits will be visible.

**3)Creating and Managing Branches**

* Write the commands to stash your changes, switch branches, and then apply the stashed changes.

**Step1:**



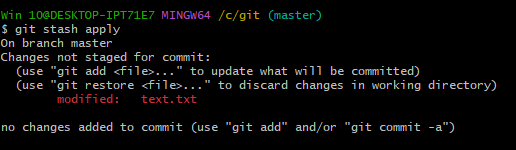
* Moving to the feature branch and have made changes in the text.txt file using the commands git checkout feature-branch and vi text.txt for changing the branch and editing the file respectively.
* Here we have not staged and commited the changes in the text.txt file.

**Step2:**

D:\git_screenshots\git_03\git_02.png

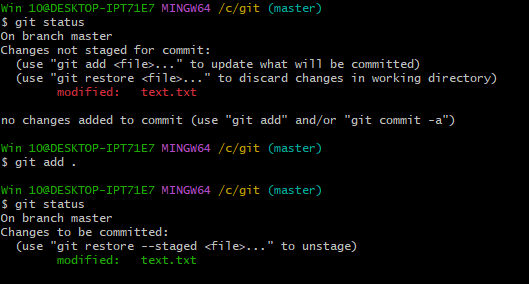
* In this step we have stashed the changes which have been made in the text.txt file in the feature-branch.
* Here we have not added/staged the file and commited the changes.
* The changes will be saved in the branch without the commiting the changes.
* The command used fir stashing the changes git stash

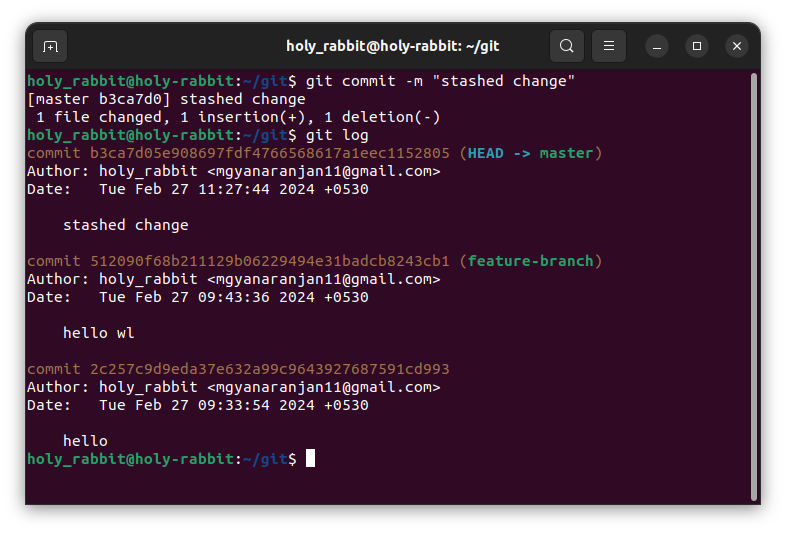
**Step3:**



* Before applying the changes made in the feature-branch first we moved to the master branch.
* Then ,in the master branch we applied the stashed changes made in the feature-branch.
* After applying the stash to the master. It will give us a message saying that the applied stash is not staged and commited in the master branch.

**Step4:**



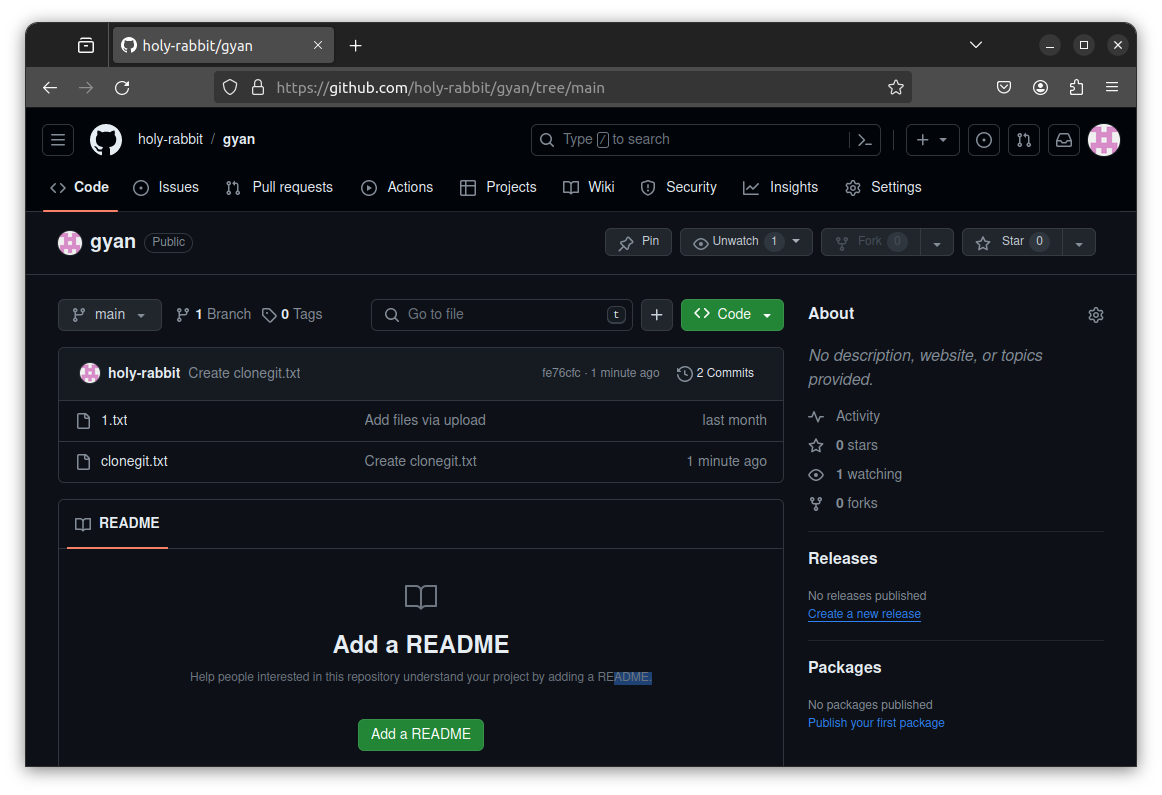


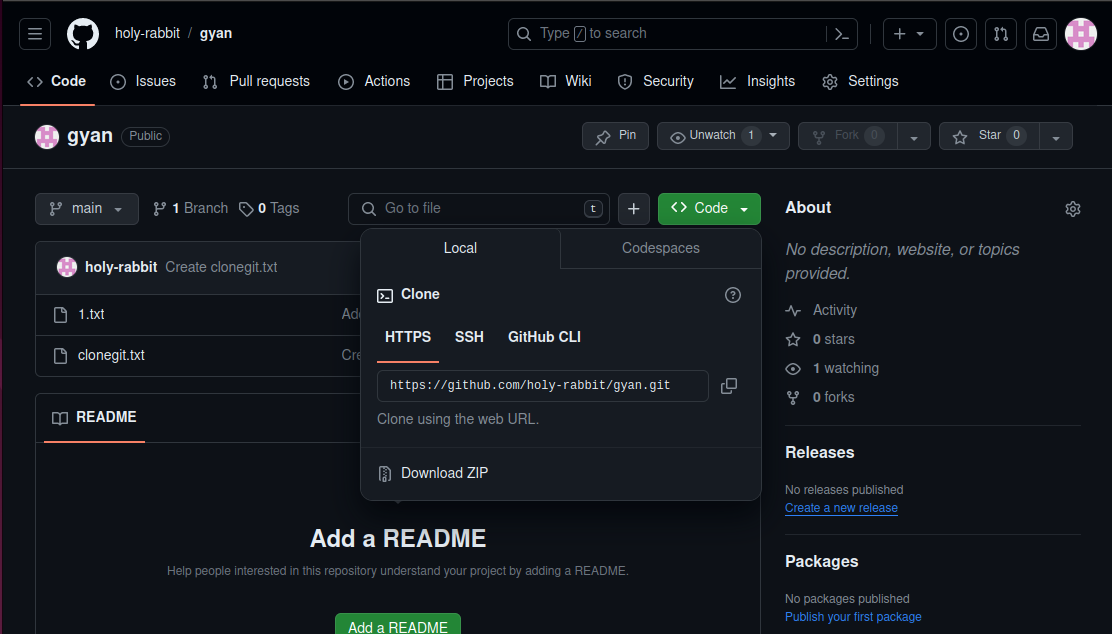
* After stashing we can see the staus ,here it showed that the file is modified but not yet committed.
* If we watch git status after adding on the stage but not commiting , then it will “say changes to be commited.”
* After commiting the changes we can see the log of the repository.

**4)Collaboration and Remote Repositories**

* Clone a remote Git repository to your local machine.

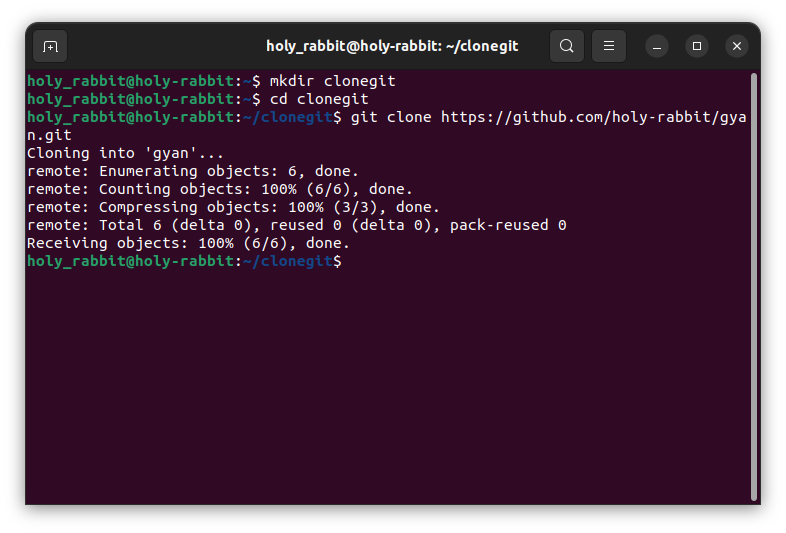
**Step1:**

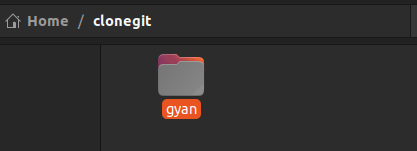




* To clone a remote git repository ,first we need to open the github.com and open any of the account on the github.com
* After that we have chosen the repository which we want to clone into our local machine.
* After choosing the repository , in that repository we clicked on the green button “code” ,which open a dropdown list of links in that ,we copied the “HTTPS” link from that. <https://github.com/memanju2005/flipkart_clone.git>

**Step2:**



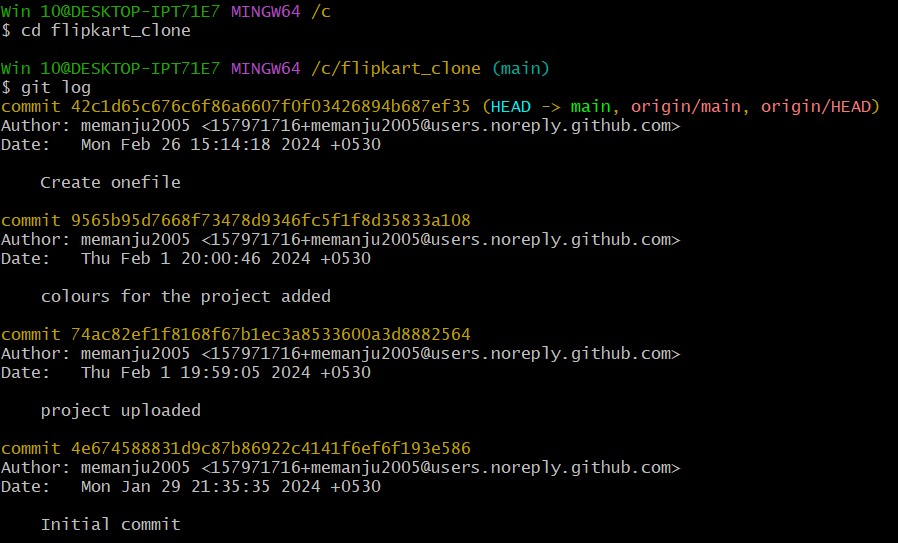


* In the second step we need to open our git bash and in some directory where you need to clone the remote repository we need to move to that location using “cd <path> “ command.
* Here we want to copy the repository to the folder clone in the c so we moved to that location
* git clone <https://github.com/memanju2005/flipkart_clone.git>  this command will copy the repository from remote to the local machine in the working directory
* you can see above the “flipkart\_clone” repository is succesfully copied in the clone directory/folder.
* We know that whole repository have been cloned to the local machine so the files and “.png” files can be seen in that repo.

**5)Collaboration and Remote Repositories**

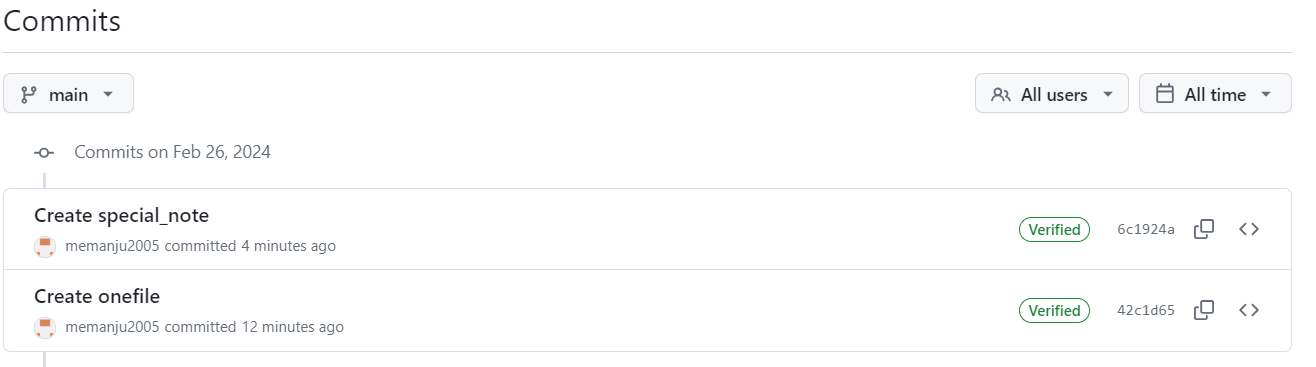
* Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.

**Step1)**



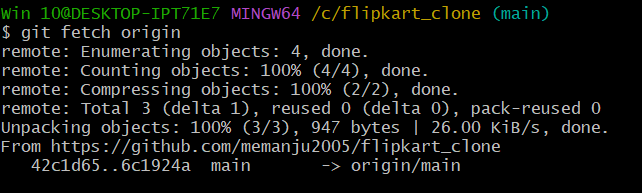
* To fetch and rebase the remote repository to local repository ,we will move to the already cloned repo.
* Initially before fetching the changes from the remote repo the last commit was “created onefile” after logging the commits

**Step2:**

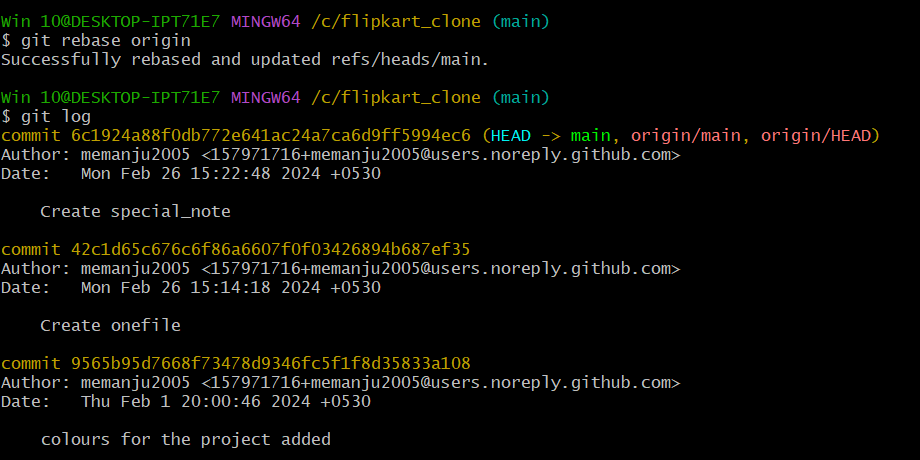


* Move to the remote repo and make some changes/add new file and commit it.

**Step3:**



* git fetch origin  this will fetch the latest changes from the remote repo that is the file named “special\_note” which was created and committed.
* These changes after fetching will not be be available in the working directory.

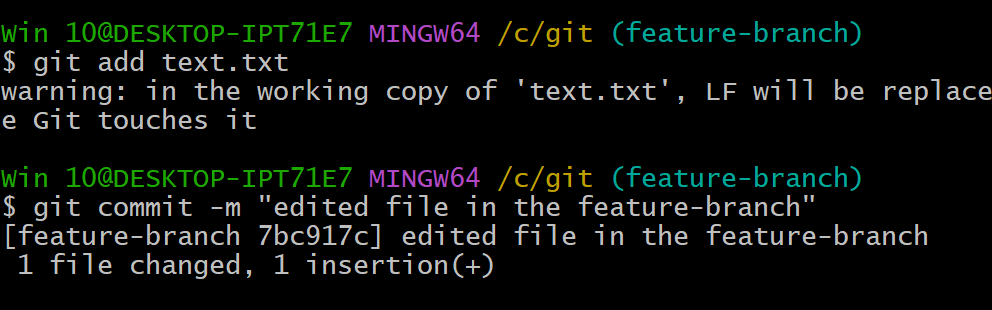
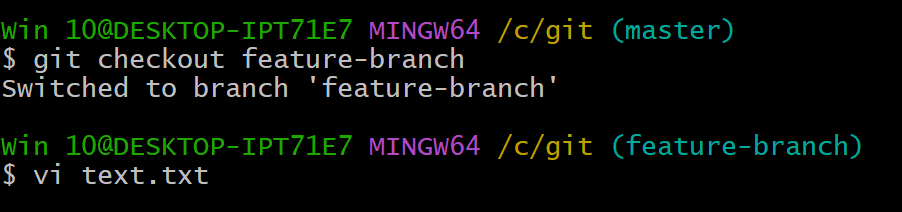
**Step4:**

* git rebase origin this command is used to bring the changes which are fetched and present in the local repo to the working directory
* after rebasing the remote branch to local branch ,the commit which are made in remote repo that will added to local branch.

**6)Collaboration and Remote Repositories**

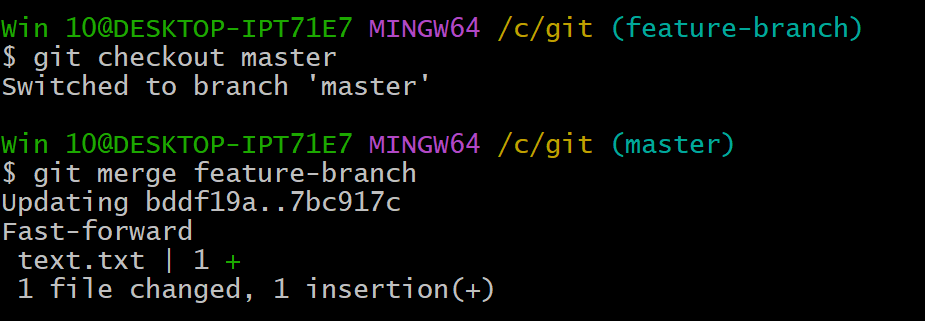
* Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge.

**Step1:**

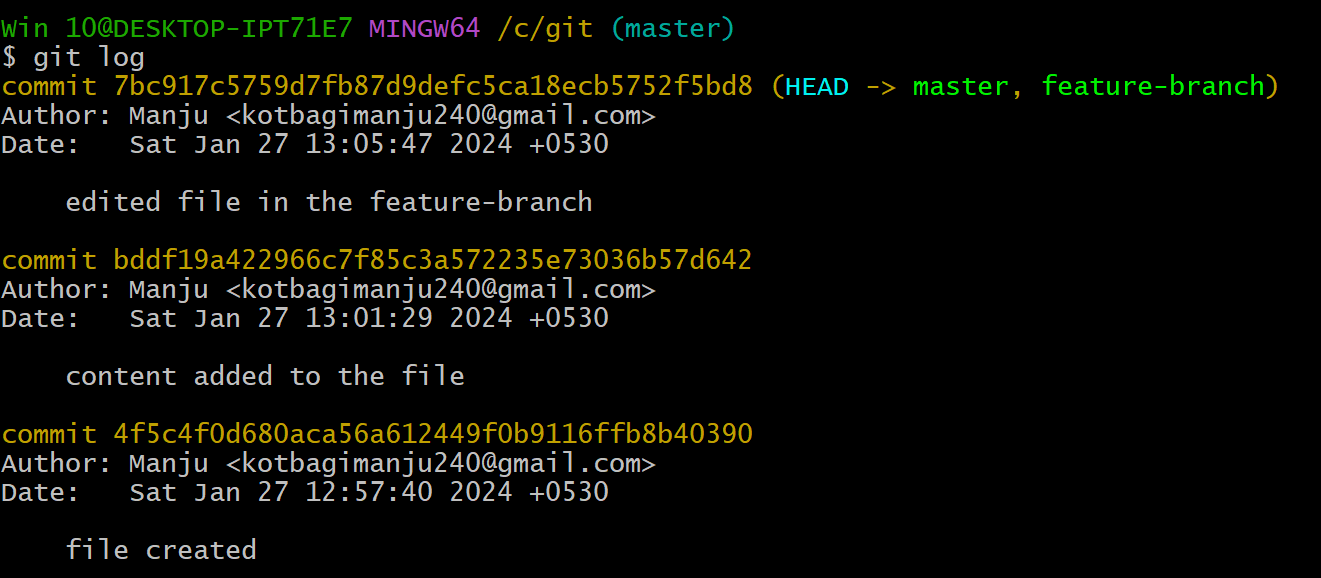


* Move to the feature branch and make some changes in the that branch ,stage and commit the changes.
* For committing we can use additional/optional  -m “message” , will commit the state with appropriate message.

**Step2:**



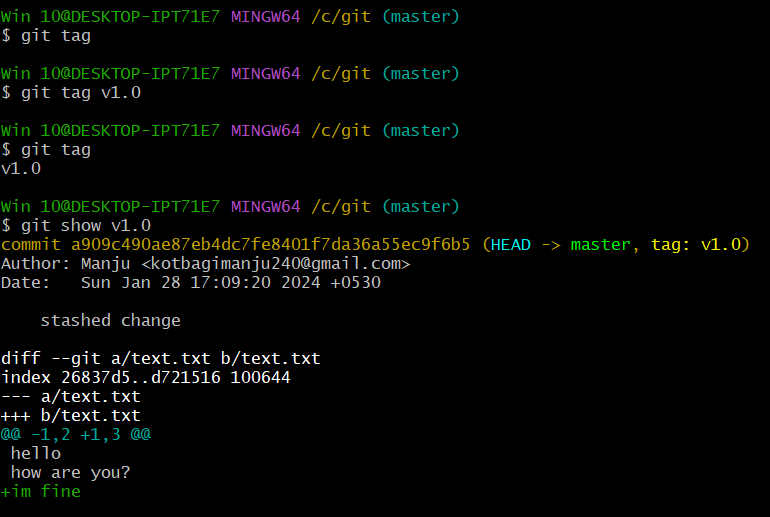
* Then checkout to the master branch and merge the branch



**7)Git Tags and Releases**

* Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.

**Step1:**

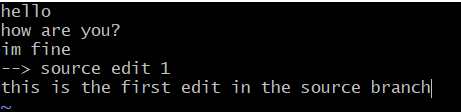
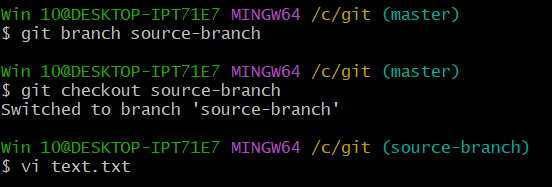


* git tag v1.0  this will create a tag of the latest commit(or we can specify the particular commit with commit ID) or we can also add a tag message using  -m “message”.
* git tag this command will show the all tags made i.e, v1.0 created.
* git show v1.0  this will show details in that tag(v1.0) with full description.

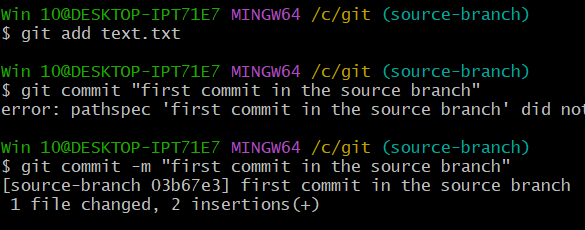
**8) Advanced Git Operations**

* Write the command to cherry-pick a range of commits from "source-branch" to the current branch.

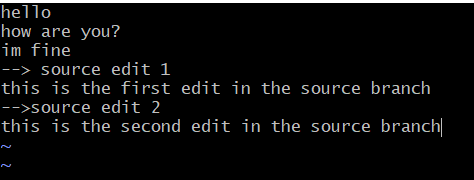
**Step1:**



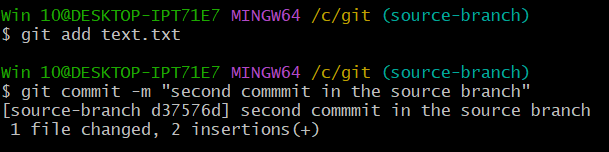
* Create a branch named source branch and check out to the source branch.
* And make the first some change in the text.txt file.



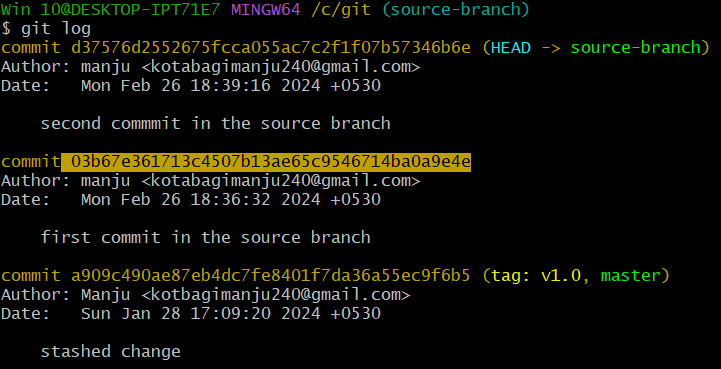
* Stage and commit the changes with commit message saying “first commit in the source branch.



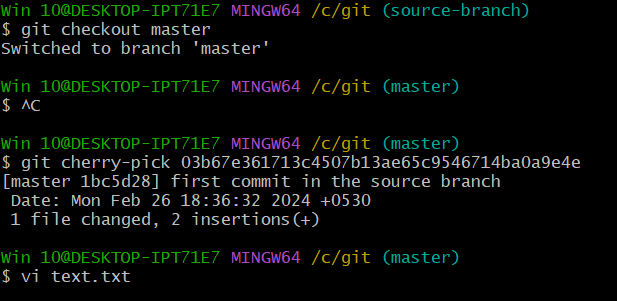
* Again make some changes in the file or add few more line in the text.txt file.



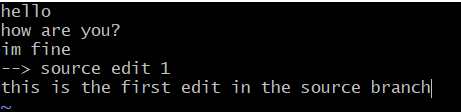
* Stage and commit the changes with message saying “second commit in the source branch”.



* Here we want copy the commit ID to cherry-pick the specific state from the git log



* Now move to the master branch.
* Git cherry-pick <commit ID> this will take the mentioned commit ID stage and merge to the master branch.
* Main advantage of using cherry pick is we can pick the required snapshot from the branches and add to the master branch.

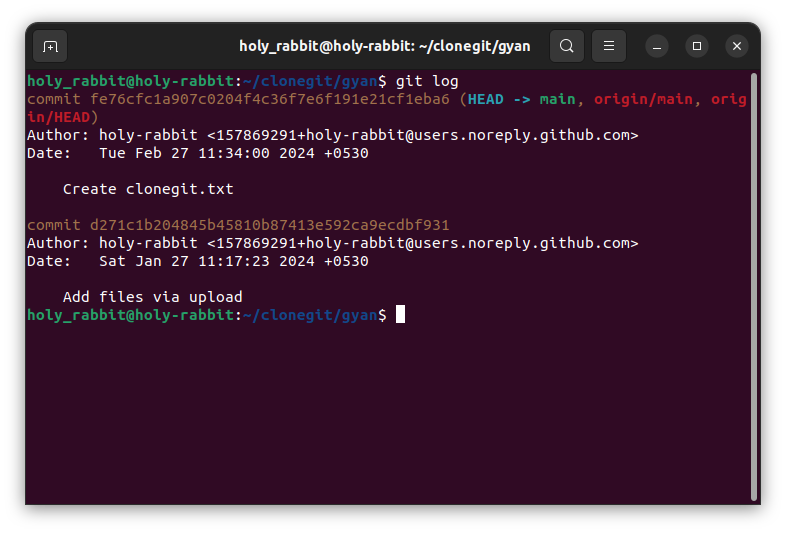


* Here we can see the content of the text.txt file at that snapshot is added to the master.

**9)Analysing and Changing Git History**

* Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?

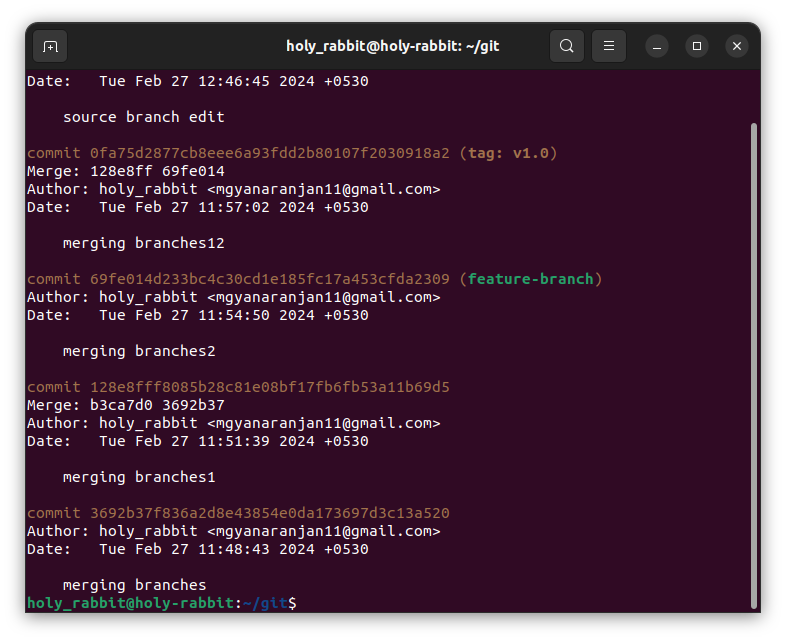
**Step1:**



* To view the details of the specific commit including author, date and commit message we should copy the specific commit which you want to view in detail.
* git show <commit ID>  this will show the full detail of the commit ID mentioned ,added changes will be shown in green colour and deleted changes will be shown in red colour.

**10)Analysing and Changing Git History**

* Write the command to list all commits made by the author "JohnDoe" between "2024-01-27" and "2023-01-28."

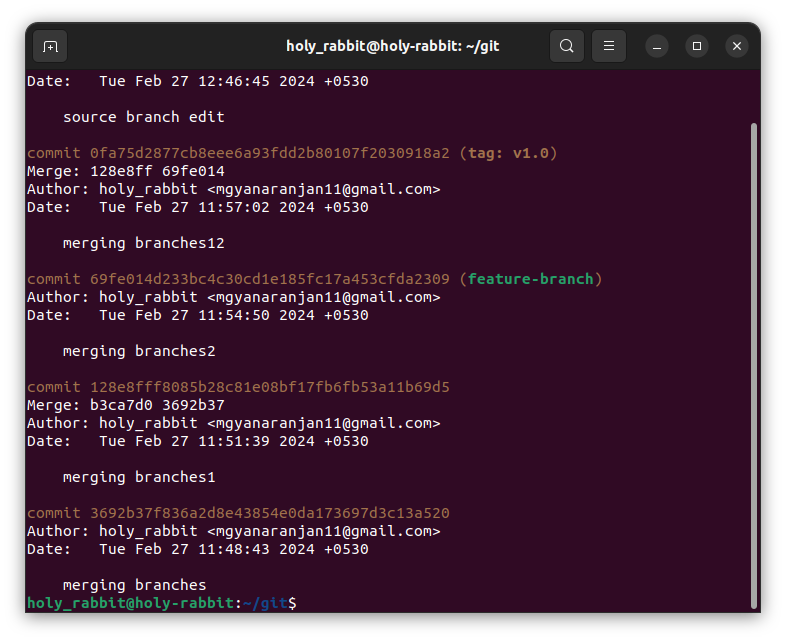


* git log --author="Manju" --since="2024-01-26" --until="2024-01-28" this will show all the commits made by the author “Manju” b/w dated “2024-01-26" and "2024-01-28".

**11)Analysing and Changing Git History**

* Write the command to display the last five commits in the repository's history.

**Step 1:**

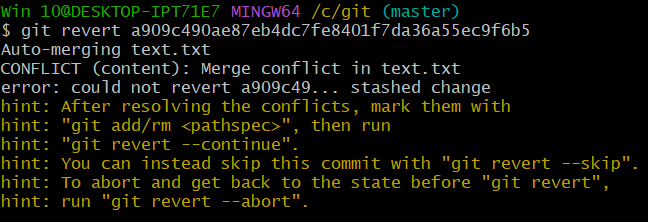


* git log –n this will display last n no.of commits. Here n is 5.

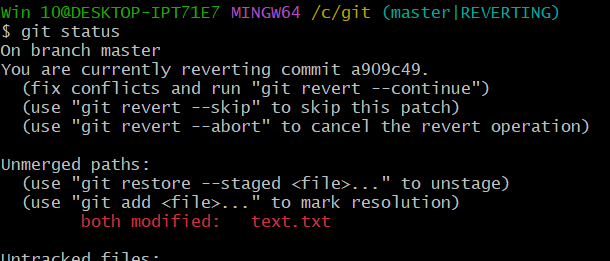
**12)Analysing and Changing Git History**

* Write the command to undo the changes introduced by the commit with the ID "abc123".

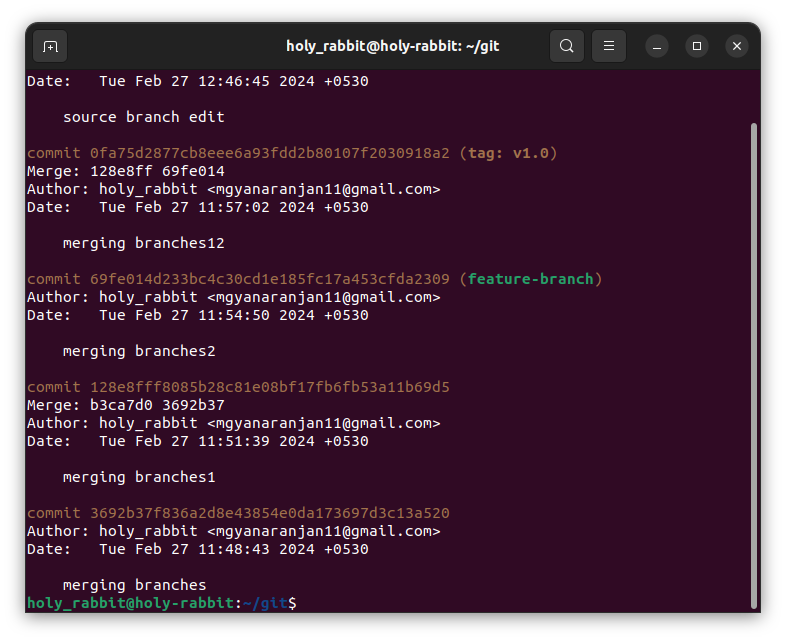
**Step 1:**



* git revert <commit ID > this will revert to the that stage of commit
* In case of failed of auto conflict , conflict will arise and we should solve conflict.



* After git status we can see the modified file “text.txt” which we had modified the file to solved the conflict.



* We should stage the file and commit with the appropriate message