SOURCE CODE MANAGEMENT

Practical File

CHITKARA UNIVERSITY

Submitted by

Name: Chayank Das

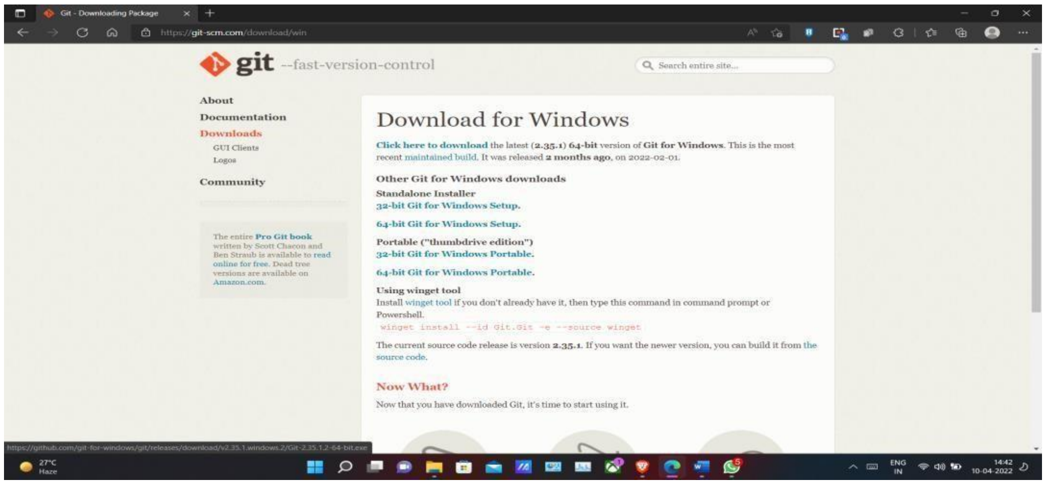
Roll no.: 2110990388

TASK 1.1

Experiment 1: Setting Up Git Client Installing

Git on windows:

1. Browse to the official Git website: https://git-scm.com/downloads
2. 2. Click the download link for Windows and allow the download to complete.



Git - Downloading Package

https://git-scm.com/download/win

git --fast-version-control

Search entire site...

About
Documentation
Downloads
GUI Clients
Logos
Community

The entire Pro Git book written by Scott Chacon and Ben Straub is available to read online for free. Dead tree versions are available on Amazon.com.

Download for Windows

Click here to download the latest (2.35.1) 64-bit version of Git for Windows. This is the most recent maintained build. It was released 2 months ago, on 2022-02-01.

Other Git for Windows downloads
Standalone Installer
32-bit Git for Windows Setup.
64-bit Git for Windows Setup.
Portable ("thumbdrive edition")
32-bit Git for Windows Portable.
64-bit Git for Windows Portable.
Using winget tool
Install winget tool if you don't already have it, then type this command in command prompt or Powershell.

```
winget install --id Git.Git -e --source winget
```

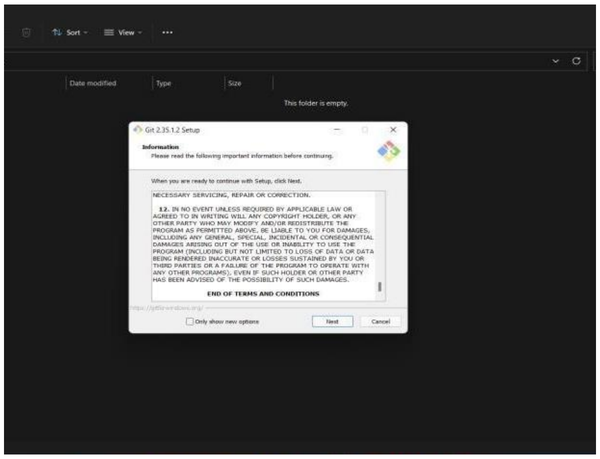
The current source code release is version 2.35.1. If you want the newer version, you can build it from the source code.

Now What?
Now that you have downloaded Git, it's time to start using it.

https://github.com/git-for-windows/git/releases/download/v2.35.1.windows.2/Git-2.35.1.2-64-bit.exe

27°C Haze ENG IN 14:42 10-04-2022

3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.
4. Allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.
5. Review the GNU General Public License, and when you're ready to install,click **Next**.



Git 2.35.1.2 Setup

Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

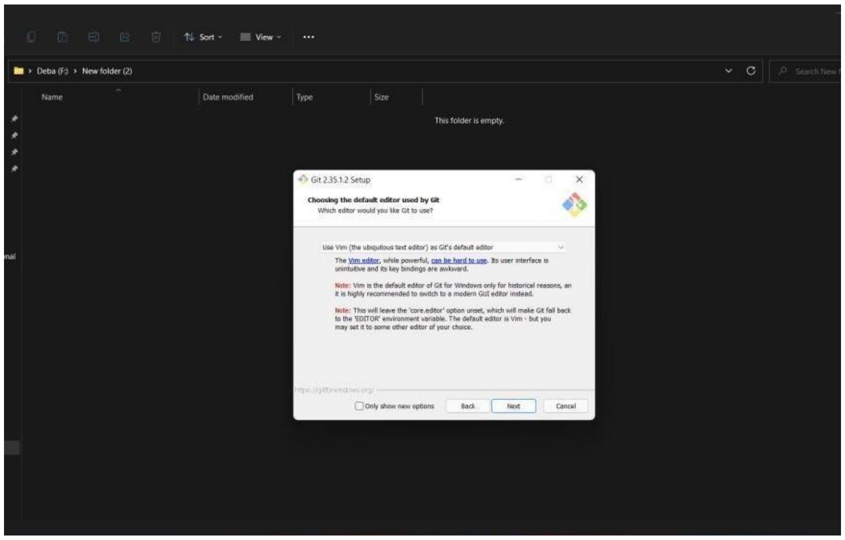
https://gitforwindows.org/

6. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.

7. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.

8. The installer will offer to create a start menu folder. Simply click **Next**.

9. Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.



Git 2.35.1.2 Setup

Choosing the default editor used by Git

Which editor would you like Git to use?

Use Vim (the ubiquitous text editor) as Git's default editor

The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.

Note: Vim is the default editor of Git for Windows only for historical reasons, an it is highly recommended to switch to a modern GUI editor instead.

Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.

https://gitforwindows.org/

10. The next step allows you to choose a different name for your initial branch. The default is 'master.' Unless you're working in a team that requires a different name, leave the default option and click **Next.**

11. This installation step allows you to change the **PATH environment**. The **PATH** is the default set of directories included when you run a command from the command line. Leave this on the middle (recommended) selection and click **Next**.

12. The installer now asks which SSH client you want Git to use. Git already comes with its own SSH client, so if you don't need a specific one, leave the default option and click **Next.**

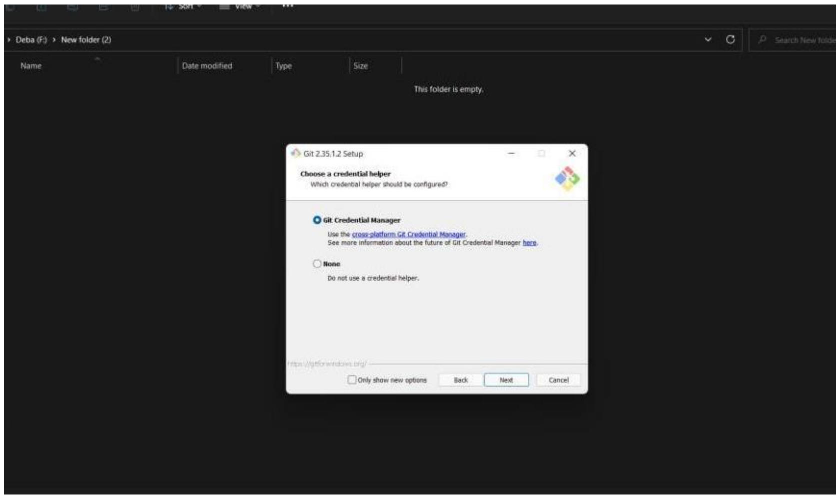
13. The next option relates to server certificates. Most users should use the default. If you're working in an Active Directory environment, you may need to switch to Windows Store certificates. Click **Next**.

14. The next selection converts line endings. It is recommended that you leave the default selection. This relates to the way data is formatted and changing this option may cause problems. Click **Next**.

15. Choose the terminal emulator you want to use. The default MinTTY is recommended, for its features. Click **Next**.

16. The installer now asks what the **git pull** command should do. The default option is recommended unless you specifically need to change its behaviour. Click **Next** to continue with the installation.

17. Next you should choose which credential helper to use. Git uses credential helpers to fetch or save credentials. Leave the default option as it is the most stable one, and click **Next**.



Choose a credential helper
Which credential helper should be configured?

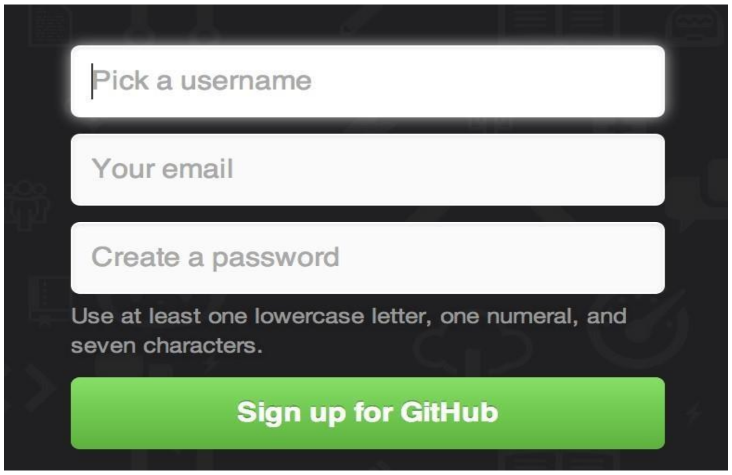
[✓ Git Credential Manager]
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.

None
Do not use a credential helper.

https://gitforwindows.org/
☐ Only show new options

Experiment 2: Setting Up Your Github Account:

The first thing you need to do is set up a free user account. Simply visit https://github.com, choose a user name that isn't already taken, provide an email address and a password, and click the big green "Sign up for GitHub" button.

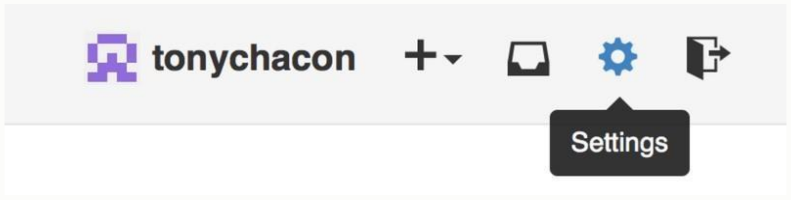

Pick a username
Your email
Create a password
Use at least one lowercase letter, one numeral, and seven characters.
Sign up for GitHub

The next thing you'll see is the pricing page for upgraded plans, but it's safe to ignore this for now. GitHub will send you an email to verify the address you provided. Go ahead and do this; it's pretty important (as we'll see later). Clicking the Octocat logo at the top-left of the screen will take you to your dashboard page. You're now ready to use GitHub.

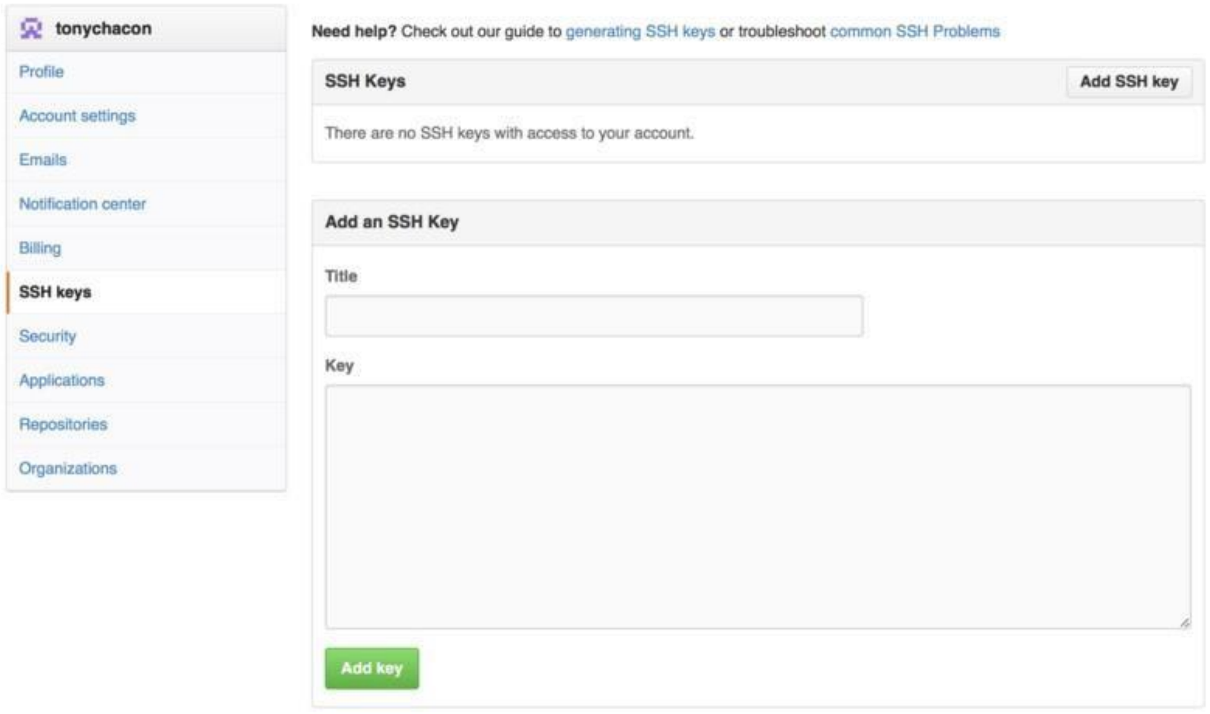
SSH Access

As of right now, you're fully able to connect with Git repositories using the protocol , authenticating with the username and password you just set up. However, to simply clone public projects, you don't even need to sign up - the account we just created comes into play when we fork projects and push to our forks a bit later.

If you'd like to use SSH remotes, you'll need to configure a public key. If you don't already have one, see Generating Your SSH Public Key. Open up your account settings using the link at the top-right of the window:



Then select the "SSH keys" section along the left-hand side.



Need help? Check out our guide to generating SSH keys or troubleshoot common SSH Problems

SSH Keys

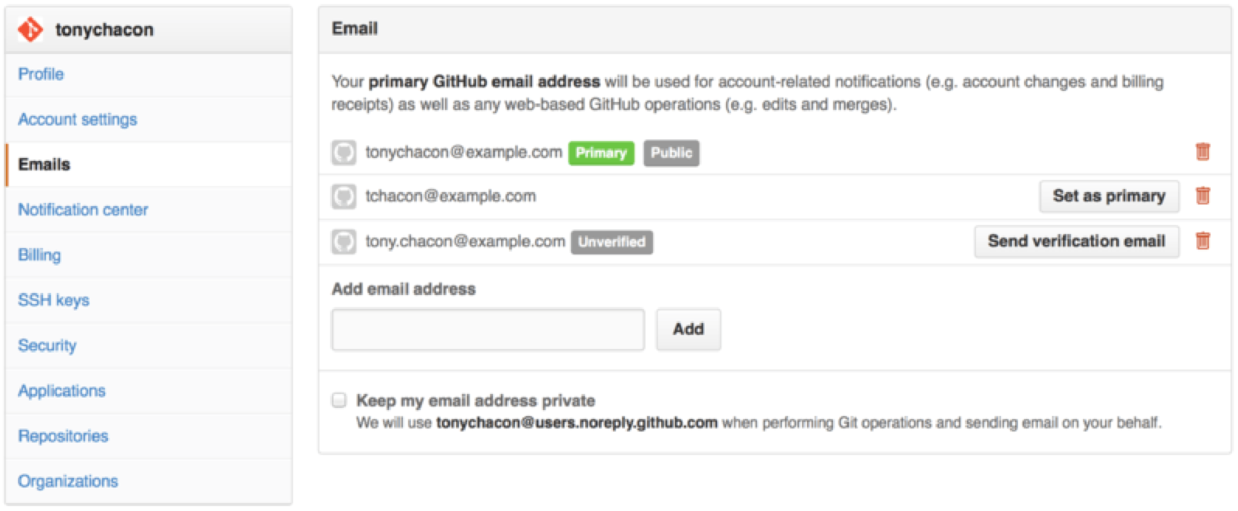
There are no SSH keys with access to your account.

Add an SSH Key

From there, click the "Add an SSH key" button, give your key a name, paste the contents of your `~/.ssh/id_rsa.pub` (or whatever you named it) public-key file into the text area, and click “Add key”.

Your Email Addresses

The way that GitHub maps your Git commits to your user is by email address. If you use multiple email addresses in your commits and you want GitHub to link them up properly, you need to add all the email addresses you have used to the Emails section of the admin section.



Your **primary GitHub email address** will be used for account-related notifications (e.g. account changes and billing receipts) as well as any web-based GitHub operations (e.g. edits and merges).

tonychacon@example.com

tchacon@example.com

tony.chacon@example.com

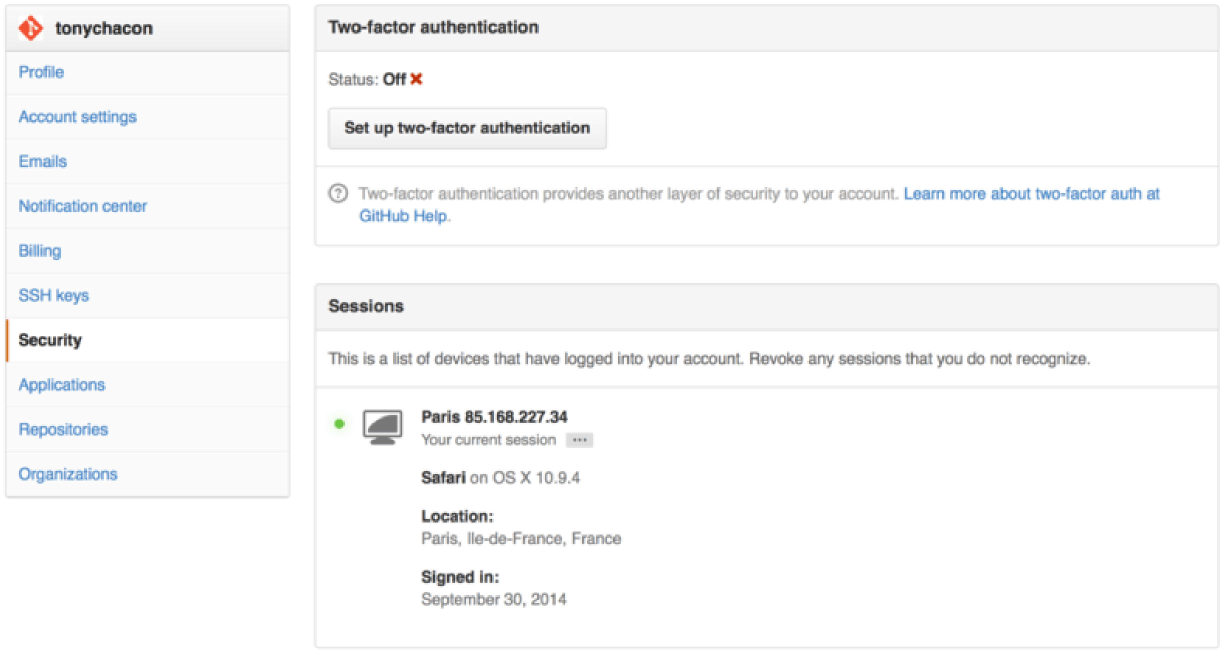
We will use **tonychacon@users.noreply.github.com** when performing Git operations and sending email on your behalf.

In Add email addresses we can see some of the different states that are possible. The top address is verified and set as the primary address, meaning that is where you’ll get any notifications and receipts. The second address is verified and so can be set as the primary if you wish to switch them. The final address is unverified, meaning that you can’t make it your primary address. If GitHub sees any of these in commit messages in any repository on the site, it will be linked to your user now.

Two Factor Authentication

Finally, for extra security, you should definitely set up Two-factor Authentication or “2FA”. Two-factor Authentication is an authentication mechanism that is becoming more and more popular recently to mitigate the risk of your account being compromised if your password is stolen somehow. Turning it on will make GitHub ask you for two different methods of authentication, so that if one of them is compromised, an attacker will not be able to access your account.

You can find the Two-factor Authentication setup under the Security tab of your Account settings.



Status: Off ✗

Two-factor authentication provides another layer of security to your account. Learn more about two-factor auth at GitHub Help.

This is a list of devices that have logged into your account. Revoke any sessions that you do not recognize.

Paris 85.168.227.34
Your current session

Safari on OS X 10.9.4

Location:
Paris, Ile-de-France, France

Signed in:
September 30, 2014

If you click on the "Set up two-factor authentication" button, it will take you to a configuration page where you can choose to use a phone app to generate your secondary code (a "time based one-time password"), or you can have GitHub send you a code via SMS each time you need to log in.

After you choose which method you prefer and follow the instructions for setting up 2FA, your account will then be a little more secure and you will have to provide a code in addition to your password whenever you log into GitHub.

Creating a repository:

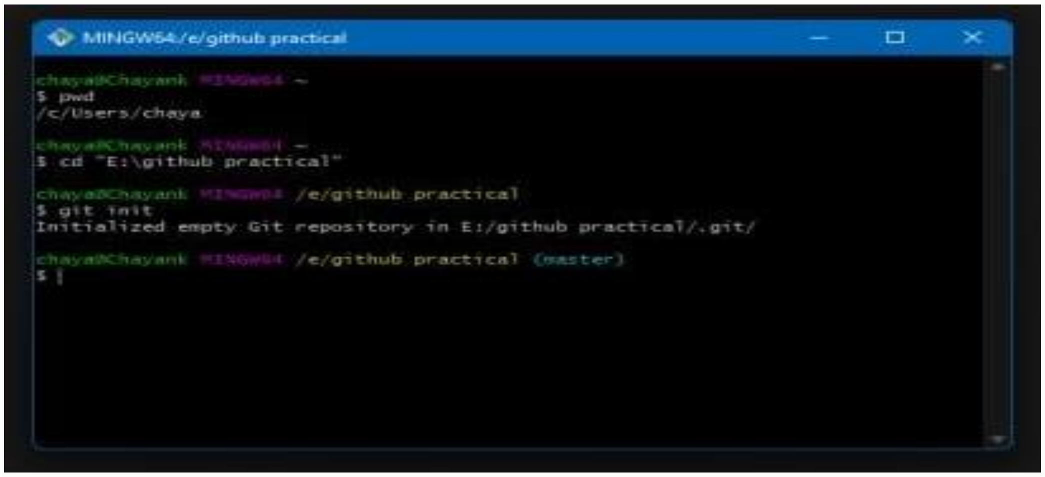
You typically obtain a Git repository in one of two ways:

1.You can take a local directory that is currently not under version control, and turn it into a Git repository, or

2.You can clone an existing Git repository from elsewhere.

A project directory that is currently not under version control and we want to

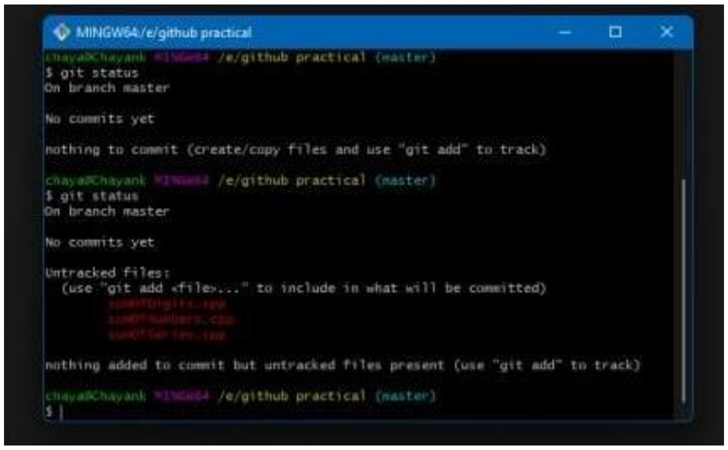
start controlling it with Git, we first need to go to that project’s directory. And we need to execute the command (git init) This creates a new subdirectory named .git that contains all of your necessary repository files — a Git repository skeleton



```
chaya@Chayank MINGW64 ~
$ pwd
/c/Users/chaya

chaya@Chayank MINGW64 ~
$ cd "E:\github practical"

chaya@Chayank MINGW64 /e/github practical
$ git init
Initialized empty Git repository in E:/github practical/.git/

chaya@Chayank MINGW64 /e/github practical (master)
$ |
```

Git Status:

The git status command 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. Status output does not show you any information regarding the committed project history. Its command is (git status)



MINGW64:/e/github practical

```
chaya@Chayank MINGW64 /e/github practical (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

chaya@Chayank MINGW64 /e/github practical (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        sumOfDigits.cpp
        sumOfNumbers.cpp
        sumOfSeries.cpp

nothing added to commit but untracked files present (use "git add" to track)

chaya@Chayank MINGW64 /e/github practical (master)
$ |
```

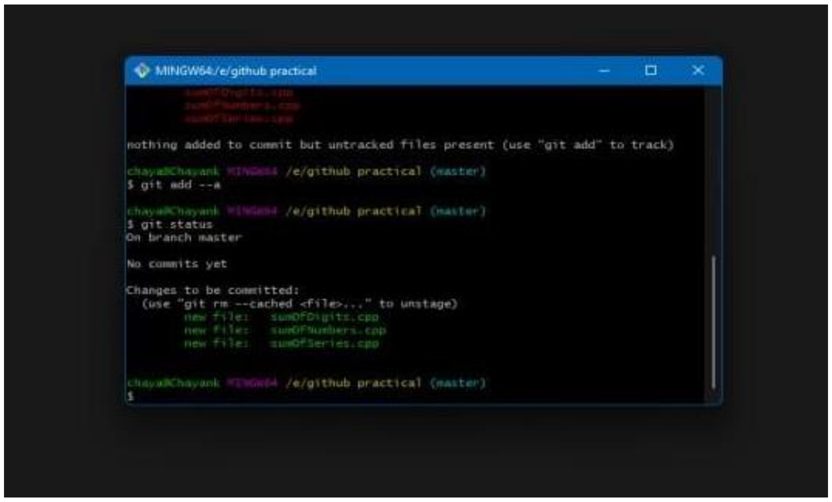
.

Git add:

1.To begin tracking a file, first we need to stage the file, to do so we use the command. (git add "name of the file")

2.To check if your file is being tracked and staged to be committed, we use the (git status) command.

3.To ensure that the file is being tracked, the file name will appear in green text, and files that are not tracked will appear in red text.



MINGW64:/e/github practical

```
        sumOfDigits.cpp
        sumOfNumbers.cpp
        sumOfSeries.cpp

nothing added to commit but untracked files present (use "git add" to track)

chaya@Chayank MINGW64 /e/github practical (master)
$ git add --a

chaya@Chayank MINGW64 /e/github practical (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   sumOfDigits.cpp
        new file:   sumOfNumbers.cpp
        new file:   sumOfSeries.cpp


chaya@Chayank MINGW64 /e/github practical (master)
$
```

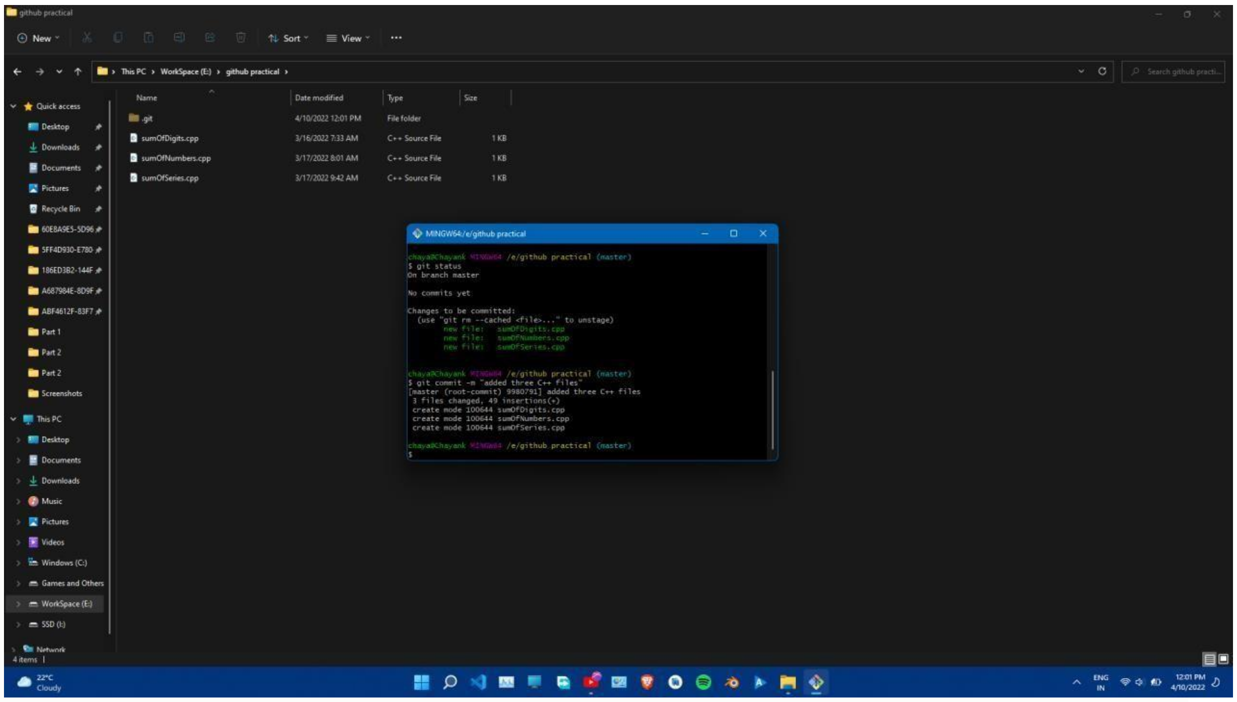
Staging modified files:

If we modify a file that is already being tracked, and run the (git status) command, the file will appear under a section named “Changes not staged for commit” which means that the file that is tracked has been modified in the working directory but not yet staged. To stage it, we run the (git add) command.

If we make changes after staging a file, and run git status, the file will again appear in the “Changes not staged for commit” section, even after staging, so to prevent it we need to stage the file after every changes we made to it.

Committing Your Changes:

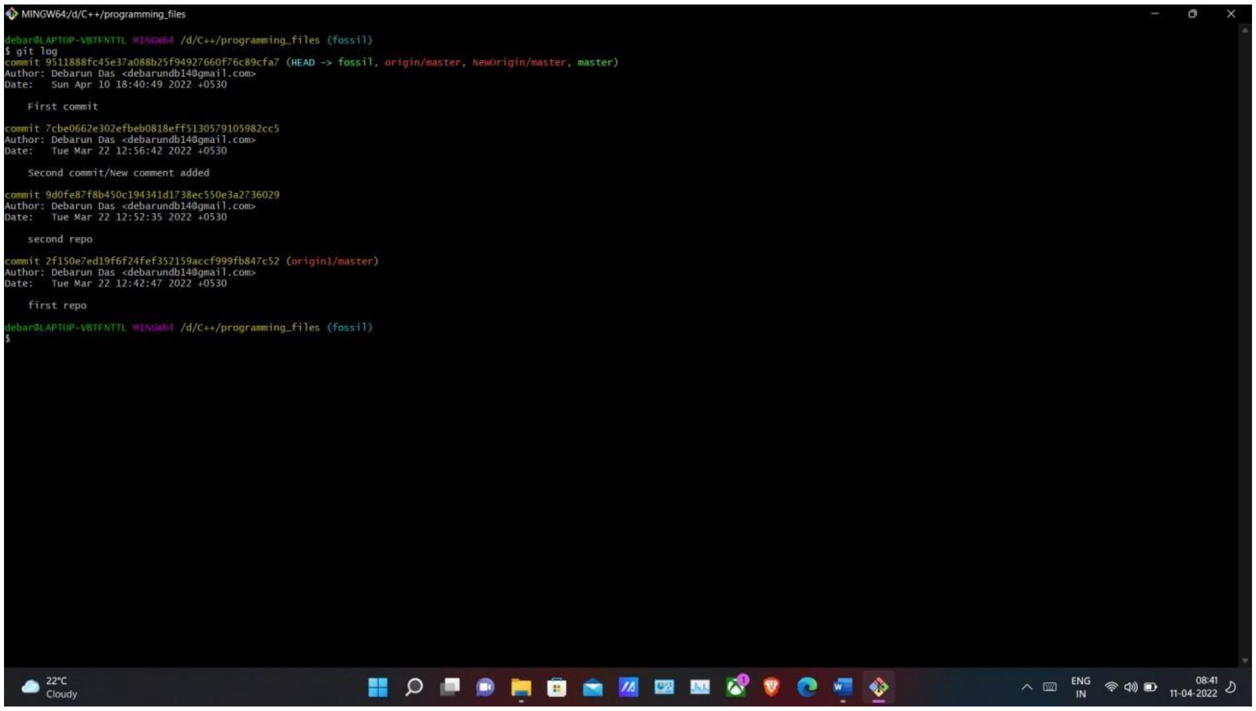
Now that our staging area is set up the way we want it, we can commit our changes. Remember that anything that is still unstaged — any files we have created or modified that we haven’t run git add on since we edited them — won’t go into this commit. They will stay as modified files on our disk. In this case, let’s say that the last time we ran git status, we saw that everything was staged, so we are ready to commit our changes. The simplest way to commit is to type git commit.



```
chaya@Chayank MINGW64 /e/github practical (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   sumOfDigits.cpp
        new file:   sumOfNumbers.cpp
        new file:   sumOfSeries.cpp


chaya@Chayank MINGW64 /e/github practical (master)
$ git commit -m "added three C++ files"
[master (root-commit) 9980791] added three C++ files
 3 files changed, 49 insertions(+)
 create mode 100644 sumOfDigits.cpp
 create mode 100644 sumOfNumbers.cpp
 create mode 100644 sumOfSeries.cpp

chaya@Chayank MINGW64 /e/github practical (master)
$
```

Experiment 3: Generate Logs

Git log command is one of the most usual commands of git. It is the most useful command of git as every time you need to check the history you have to use the git log command. The basic git log command will display the most recent commits and the status of the head. It can be use as (git log)



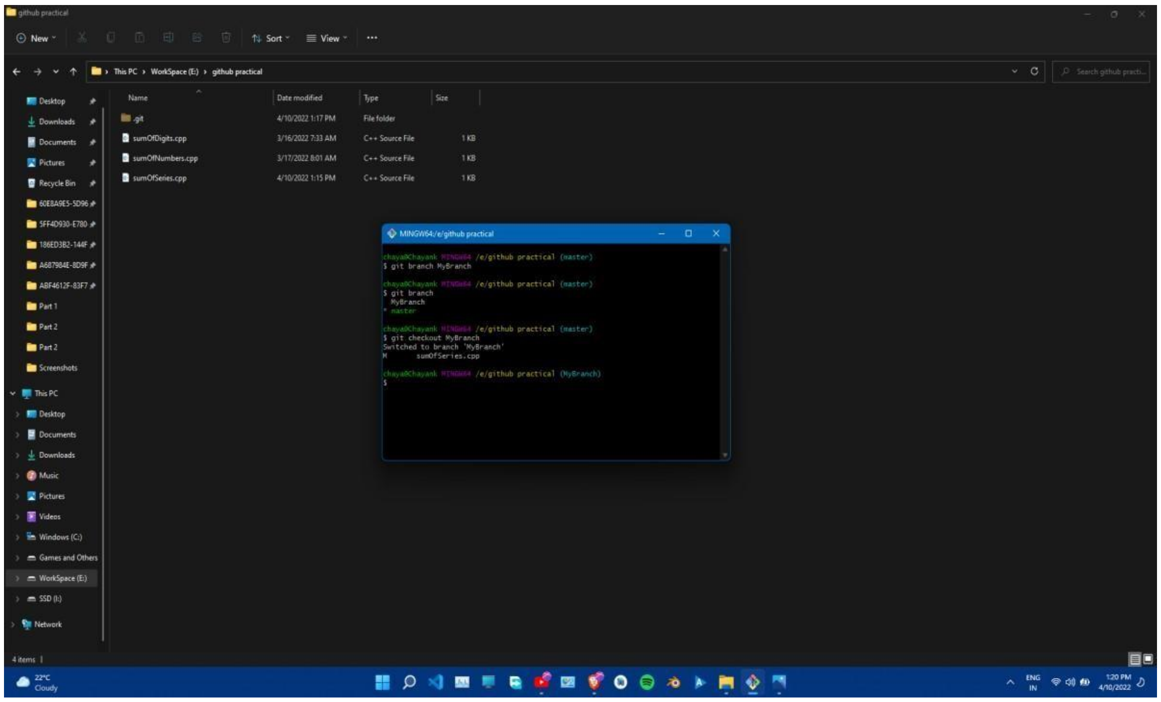
MINGW64:/d/C++/programming_files

```
debar@LAPTOP-VBTFNTTL MINGW64 /d/C++/programming_files (fossil)
$ git log
commit 9511888fc45e37a088b25f94927660f76c89cfa7 (HEAD -> fossil, origin/master, NewOrigin/master, master)
Author: Debarun Das <debarundb14@gmail.com>
Date:   Sun Apr 10 18:40:49 2022 +0530

    First commit

commit 7cbe0662e302efbeb0818eff5130579105982cc5
Author: Debarun Das <debarundb14@gmail.com>
Date:   Tue Mar 22 12:56:42 2022 +0530

    Second commit/New comment added

commit 9d0fe87f8b450c194341d1738ec550e3a2736029
Author: Debarun Das <debarundb14@gmail.com>
Date:   Tue Mar 22 12:52:35 2022 +0530

    second repo

commit 2f150e7ed19f6f24fef352159accf999fb847c52 (origin1/master)
Author: Debarun Das <debarundb14@gmail.com>
Date:   Tue Mar 22 12:42:47 2022 +0530

    first repo

debar@LAPTOP-VBTFNTTL MINGW64 /d/C++/programming_files (fossil)
$
```

Create Branches in git:

The main branch in git is called master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the files which are created in branch are not shown in master branch. We also can merge both the parent(master) and child (other branches).

1. For creating a new branch: git branch “name of branch”
2. To check how many branches we have : git branch
3. To change the present working branch: git checkout “name of the branch”



```
chaya@Chayank MINGW64 /e/github practical (master)
$ git branch MyBranch

chaya@Chayank MINGW64 /e/github practical (master)
$ git branch
  MyBranch
* master

chaya@Chayank MINGW64 /e/github practical (master)
$ git checkout MyBranch
Switched to branch 'MyBranch'
M       sumOfSeries.cpp

chaya@Chayank MINGW64 /e/github practical (MyBranch)
$
```

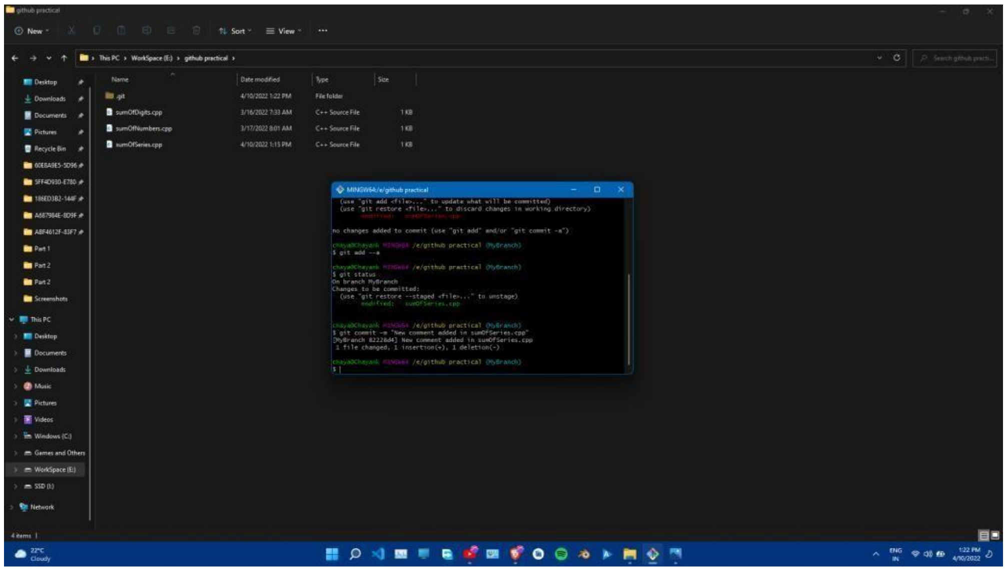
Experiment 4: Create And Visualize Branches:

To visualize, suppose we have to create a new file “hello” in the new branch “MyBranch” instead of the master branch. After this we have to do three step i.e. working directory, staging area and git repository.

After this, we have completed the 3 step process which is tracking the file, send it to stagging area and commit in the repo. Finally we can rollback to any previously saved version of this file.

Now, we will change the branch from MyBranch to master, but when we switch back to master branch the file we created i.e “hello” will not be there. Hence the new file will not be shown in the master branch. In this way we can create and change different branches. We can also merge the branches by using the git merge command to merge the other branch with the master branch. In this way we can create and change different branches.

Committing in the New Branch:

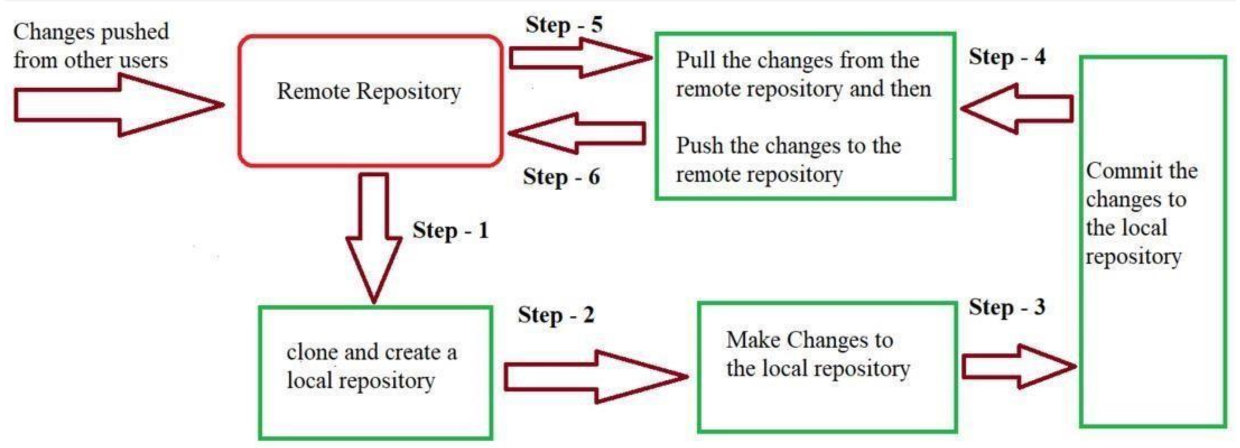


```
Name                 Date modified        Type             Size
.git                 4/10/2022 1:22 PM    File folder
sumOfDigits.cpp      3/16/2022 7:33 AM    C++ Source File  1 KB
sumOfNumbers.cpp     3/17/2022 8:01 AM    C++ Source File  1 KB
sumOfSeries.cpp      4/10/2022 1:15 PM    C++ Source File  1 KB
```

```
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   sumOfSeries.cpp

no changes added to commit (use "git add" and/or "git commit -a")

chaya@Chayank MINGW64 /e/github practical (MyBranch)
$ git add --a

chaya@Chayank MINGW64 /e/github practical (MyBranch)
$ git status
On branch MyBranch
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   sumOfSeries.cpp


chaya@Chayank MINGW64 /e/github practical (MyBranch)
$ git commit -m "New comment added in sumOfSeries.cpp"
[MyBranch 82228d4] New comment added in sumOfSeries.cpp
 1 file changed, 1 insertion(+), 1 deletion(-)

chaya@Chayank MINGW64 /e/github practical (MyBranch)
$ |
```

Experiment 5: Git Lifecycle Description :

Git is used in our day-to-day work, we use Git for keeping a track of our files, working in a collaboration with our team, to go back to our previous code versions if we face some error. Git helps us in many ways. Let us look at the Lifecycle description that git has and understand more about its life cycle. Let us see some of the basic steps that we have to follow while working with Git-



Changes pushed from other users
Remote Repository
Step - 5
Pull the changes from the remote repository and then
Push the changes to the remote repository
Step - 6
Step - 4
Commit the changes to the local repository
Step - 1
Step - 2
clone and create a local repository
Make Changes to the local repository
Step - 3

Step 1- We first clone any of the code residing in the remote repository to make our won local repository.

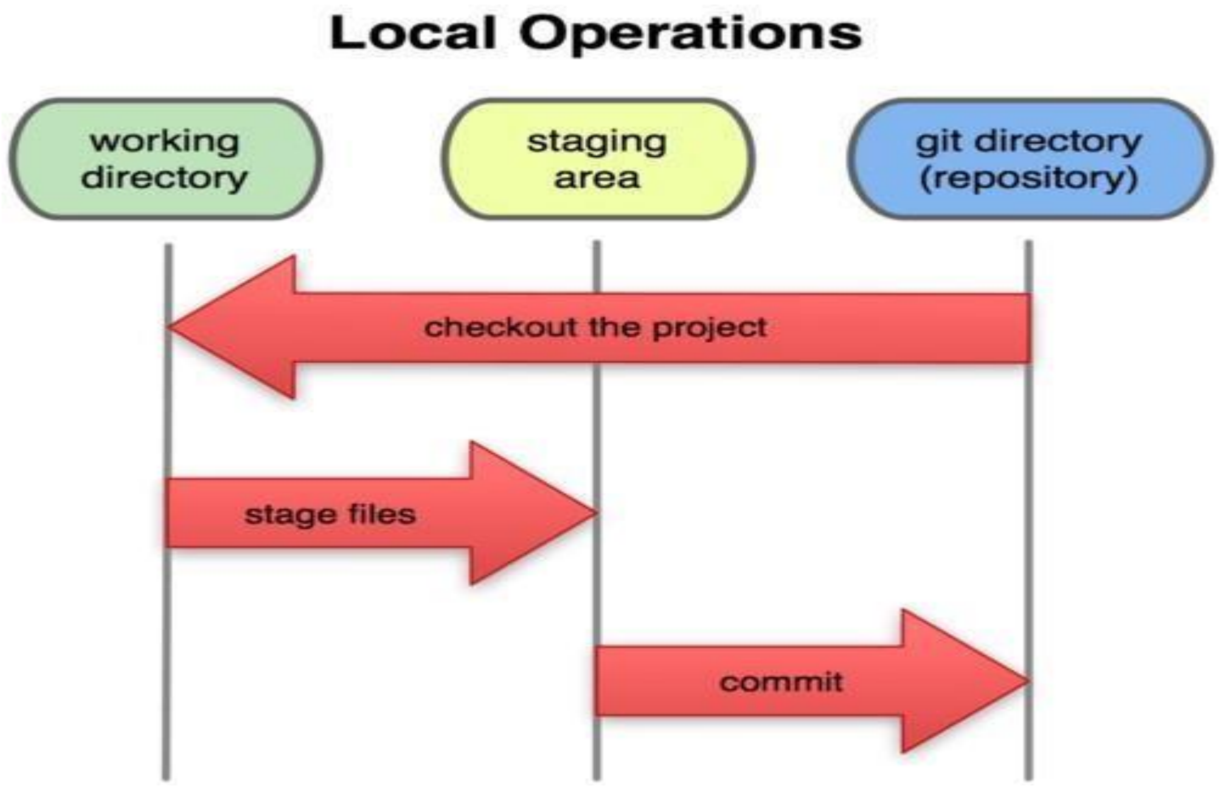
Step 2- We edit the files that we have cloned in our local repository and make the necessary changes in it.

Step 3- We commit our changes by first adding them to our staging area and committing them with a commit message.

Step 4- We first check whether there are any of the changes done in the remote repository by some other users and we first pull that changes.

Step 5- If there are no changes we push our changes to the remote repository and we are done with our work.

When a directory is made a git repository, there are mainly 3 states which make the essence of Git version Control System. The three states are:



1. Working Directory

Whenever we want to initialize aur local project directory to make a Git repository, we use the git init command. After this command, git becomes aware of the files in the project although it does not track the files yet. The files are further tracked in the staging area.

2. Staging Area

Now, to track files the different versions of our files we use the command git add. We can term a staging area as a place where different versions of our files are stored. git add command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don't want to get tracked, examples include node modules, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the .git folder inside the index file. git add git add.

3. Git Directory

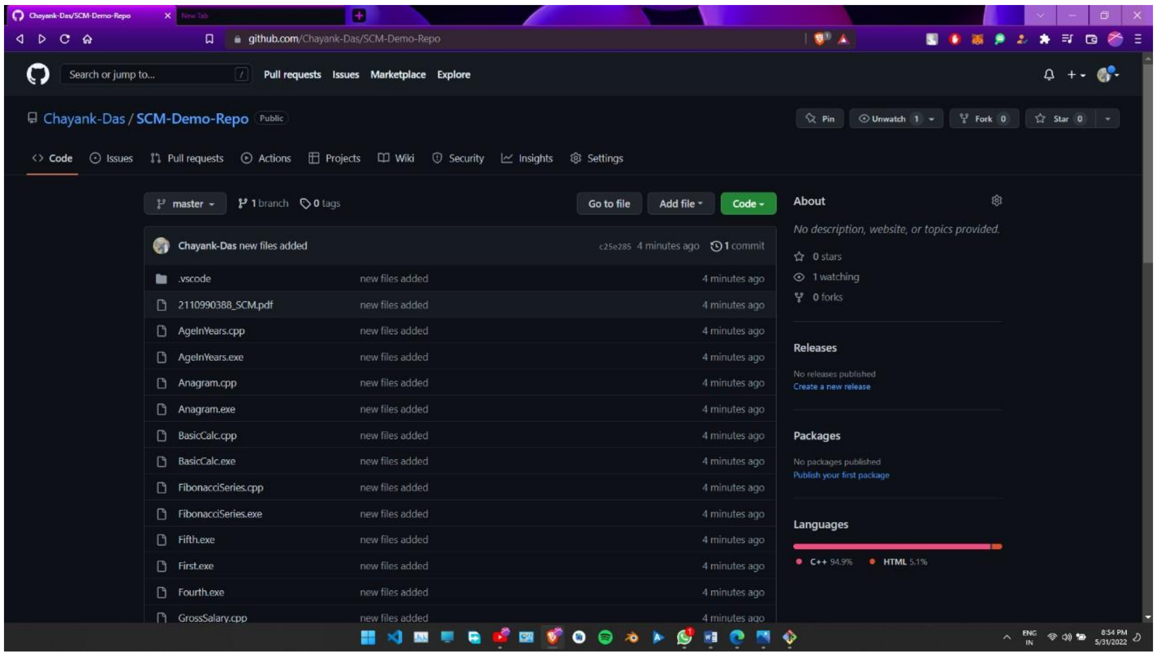
Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit aur files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message. git commit -m

TASK 1.2

Experiment 6: Add Collaborators on Github Repo :

Even if you have a public repository in GitHub, not everyone has the permission to push code into your repository. Other users have a read-only access and cannot modify the repository. In order to allow other individuals to make changes to your repository, you need to invite them to collaborate to the project. The following steps should be performed to invite other team members to collaborate with your repository:-

Step 1 – Click on the Settings tab in the right corner of the GitHub page.

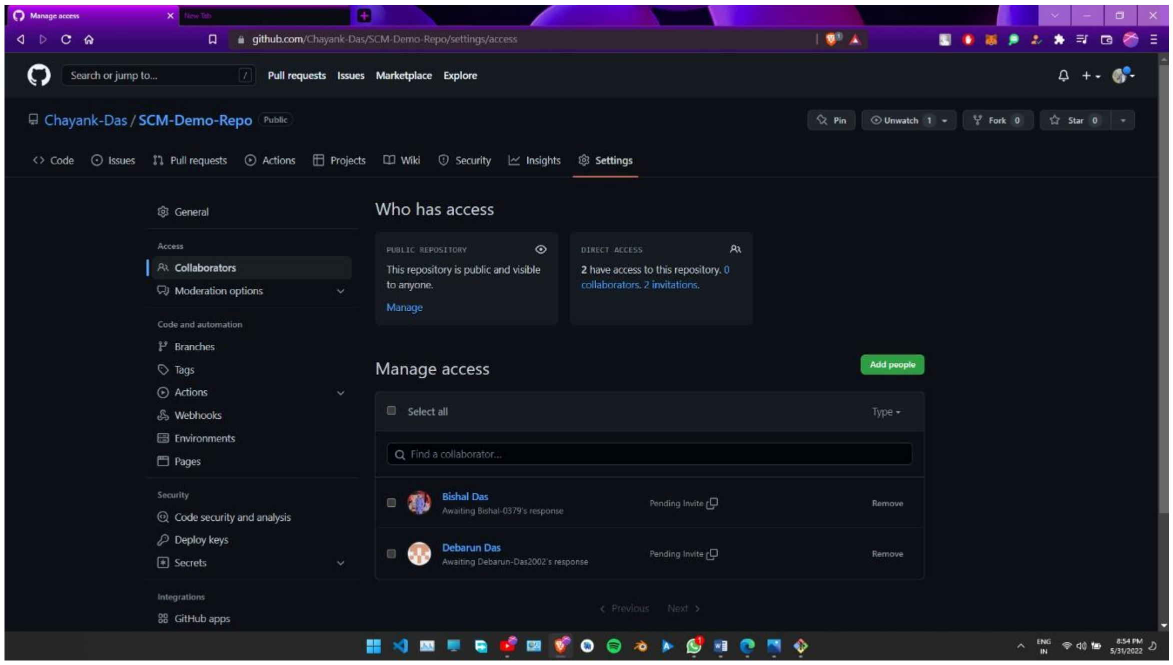

github.com/Chayank-Das/SCM-Demo-Repo
Search or jump to...
Pull requests Issues Marketplace Explore
Chayank-Das / SCM-Demo-Repo Public
Pin Unwatch 1 Fork 0 Star 0
Code Issues Pull requests Actions Projects Wiki Security Insights Settings
master 1 branch 0 tags
Go to file Add file Code
About
Releases
Packages
Languages
C++ 94.9% HTML 5.1%

Chayank-Das new files added c25e285 4 minutes ago 1 commit

.vscode	new files added	4 minutes ago
2110990388_SCM.pdf	new files added	4 minutes ago
AgeInYears.cpp	new files added	4 minutes ago
AgeInYears.exe	new files added	4 minutes ago
Anagram.cpp	new files added	4 minutes ago
Anagram.exe	new files added	4 minutes ago
BasicCalc.cpp	new files added	4 minutes ago
BasicCalc.exe	new files added	4 minutes ago
FibonacciSeries.cpp	new files added	4 minutes ago
FibonacciSeries.exe	new files added	4 minutes ago
Fifth.exe	new files added	4 minutes ago
First.exe	new files added	4 minutes ago
Fourth.exe	new files added	4 minutes ago
GrossSalary.cpp	new files added	4 minutes ago

No description, website, or topics provided.
0 stars
1 watching
0 forks
No releases published
Create a new release
No packages published
Publish your first package

Step 2 – Go to Collaborators option under the Settings tab. On the Collaborators page, you will see an Add people link as shown in the below diagram.



Who has access

PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage

DIRECT ACCESS
2 have access to this repository. 0 collaborators. 2 invitations.

Manage access

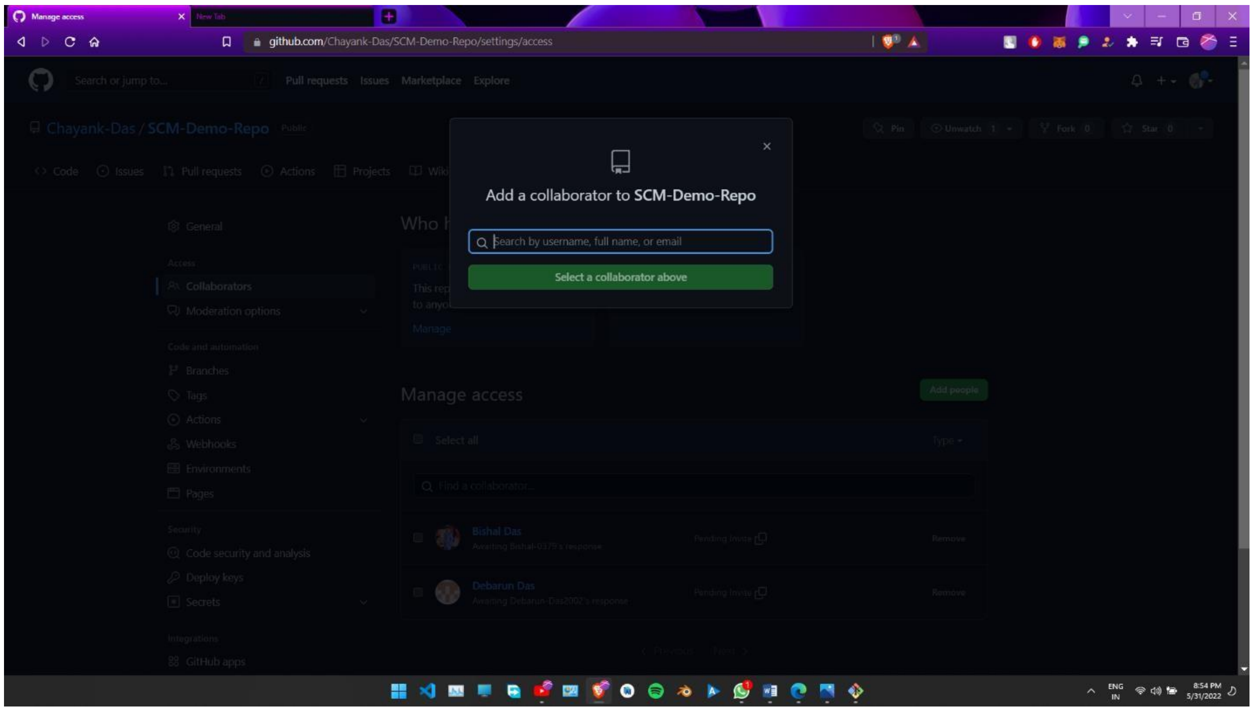
Bishal Das
Awaiting Bishal-0379's response

Debarun Das
Awaiting Debarun-Das2002's response

Step 3 – You can Invite collaborators by any of the following options –

- Username
- Full name
- Email

After you send the invite, the collaborator receives an email invitation. The collaborator has to accept it in order to get permission to collaborate on the same project.


Add a collaborator to SCM-Demo-Repo
Search by username, full name, or email
Select a collaborator above

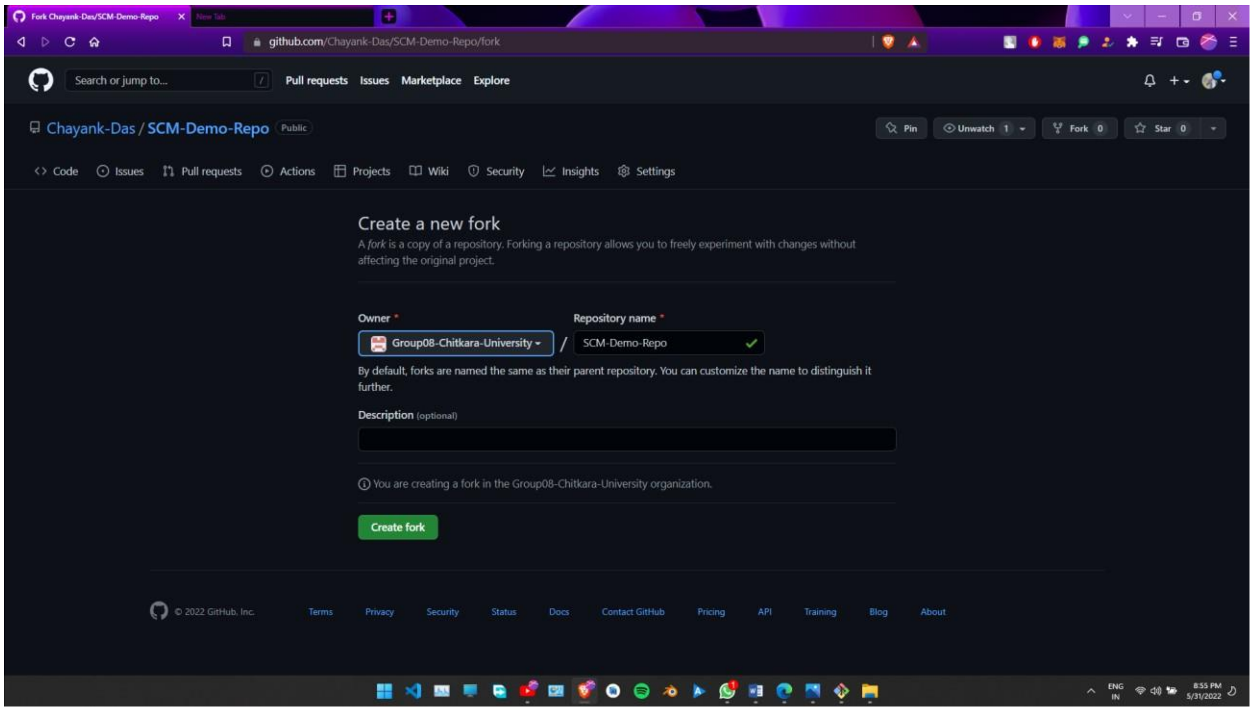
The Manage Access option also allows a repository owner to view the invitations that are pending and not accepted.

Experiment 7: Fork And Commit :

Fork is a duplicate of your original repository in which you can make the changes without affecting the original project. The following steps should be performed to fork and commit changes in your repository:-

Step 1 – To fork a project, click on the Fork button as shown below –

Step 2 – After forking the project, you need to add the forked project to a fork group by clicking on it


Chayank-Das / SCM-Demo-Repo Public

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Owner *: Group08-Chitkara-University / Repository name *: SCM-Demo-Repo

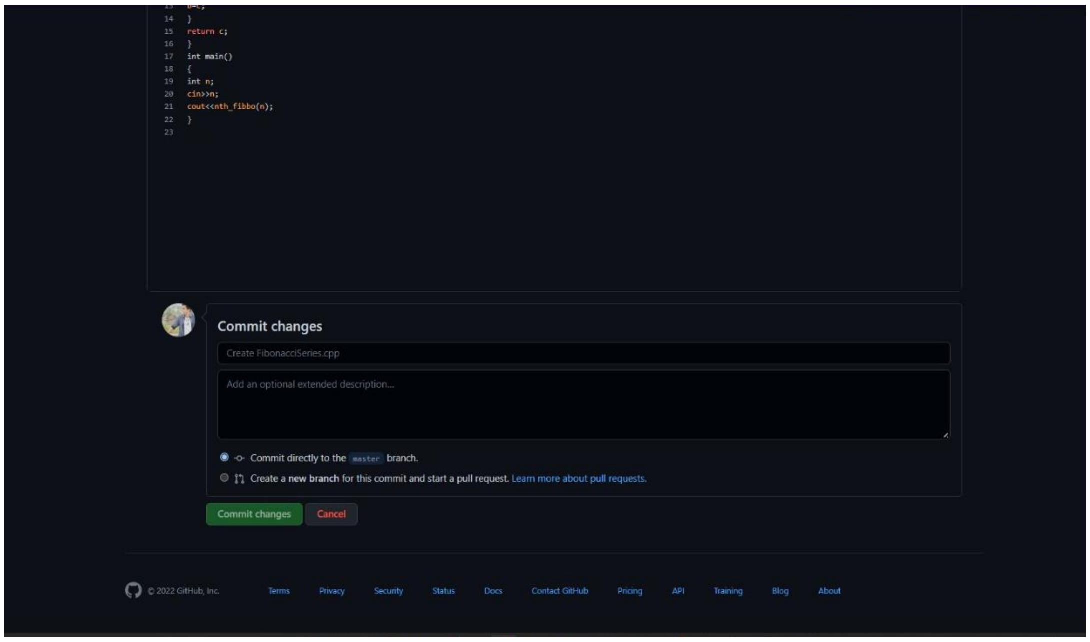
By default, forks are named the same as their parent repository. You can customize the name to distinguish it further.

Description (optional)

You are creating a fork in the Group08-Chitkara-University organization.

Step 3 – Next it will start processing of forking a project for sometime as shown below –

Step 4 – After the forking process is complete, you can commit changes in the file –



```
}
return c;
}
int main()
{
int n;
cin>>n;
cout<<nth_fibbo(n);
}
```

Commit changes

Create FibonacciSeries.cpp

Add an optional extended description...

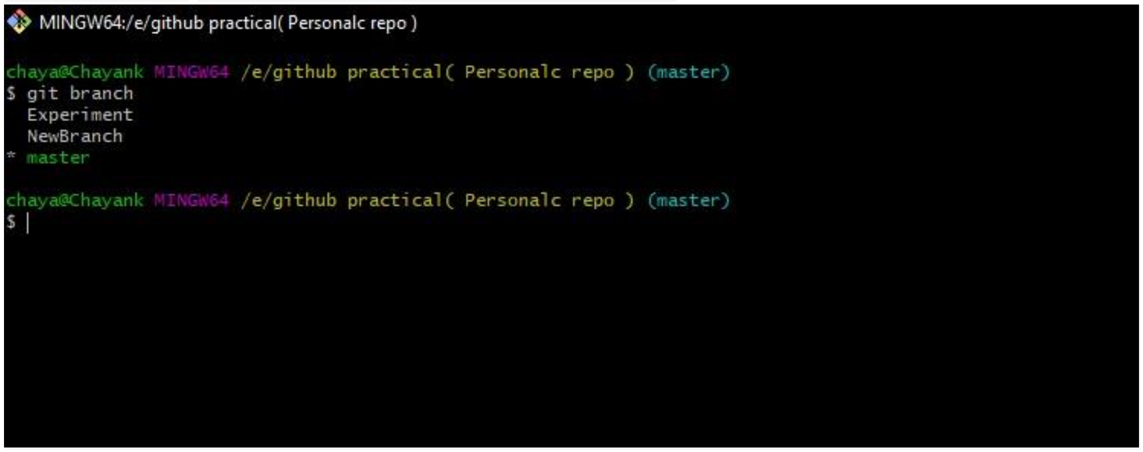
[✓ Commit directly to the master branch.]

Create a new branch for this commit and start a pull request. Learn more about pull requests.

Experiment 8: Merge And Resolve Conflicts Created Due To Own Activity And Collaborators Activity :

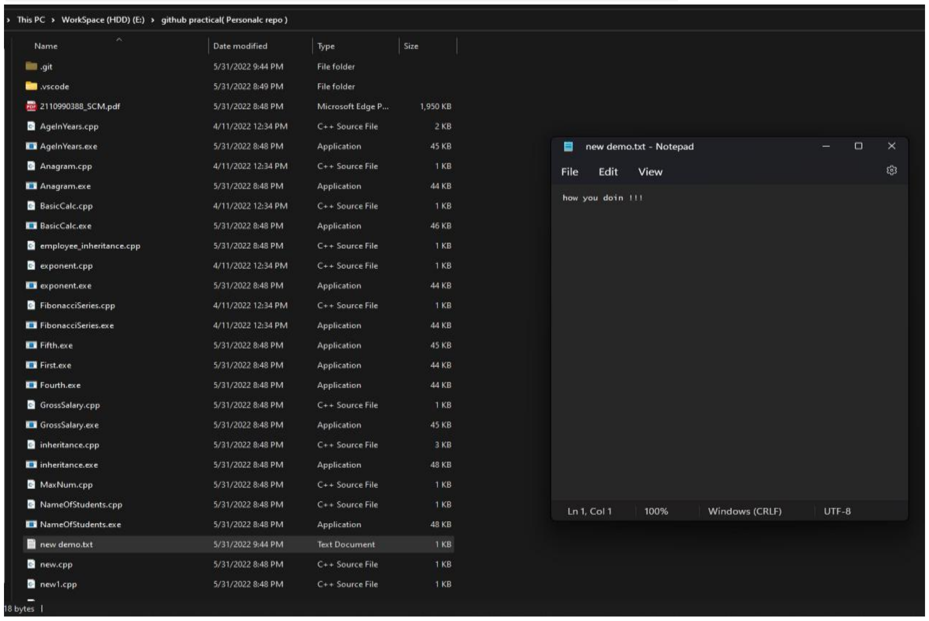
Git merge will combine multiple sequences of commits into one unified history. In the most frequent use cases, git merge is used to combine two branches into single one or in other words apply the changes to the master branch. Following steps should be performed to merge branches :-

Step 1 – Below are the branches that are created in my repo. We can see our branches by using this command **git branch.**



```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git branch
  Experiment
  NewBranch
* master

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ |
```

Step 2 – Created a new text file called new demo.txt in the master branch and inserted text 'how you doin !!!' and saved it.



```
› This PC › WorkSpace (HDD) (E:) › github practical( Personalc repo )
```

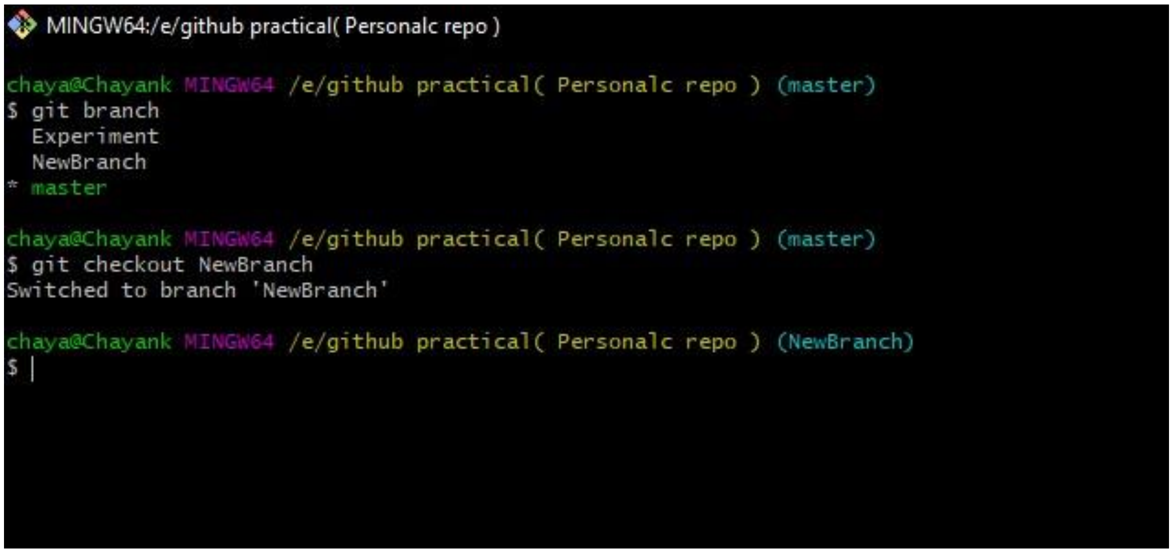
Name	Date modified	Type	Size
.git	5/31/2022 9:44 PM	File folder	
.vscode	5/31/2022 8:49 PM	File folder	
2110990388_SCM.pdf	5/31/2022 8:48 PM	Microsoft Edge P...	1,950 KB
AgeInYears.cpp	4/11/2022 12:34 PM	C++ Source File	2 KB
AgeInYears.exe	5/31/2022 8:48 PM	Application	45 KB
Anagram.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
Anagram.exe	5/31/2022 8:48 PM	Application	44 KB
BasicCalc.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
BasicCalc.exe	5/31/2022 8:48 PM	Application	46 KB
employee_inheritance.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
exponent.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
exponent.exe	5/31/2022 8:48 PM	Application	44 KB
FibonacciSeries.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
FibonacciSeries.exe	4/11/2022 12:34 PM	Application	44 KB
Fifth.exe	5/31/2022 8:48 PM	Application	45 KB
First.exe	5/31/2022 8:48 PM	Application	44 KB
Fourth.exe	5/31/2022 8:48 PM	Application	44 KB
GrossSalary.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
GrossSalary.exe	5/31/2022 8:48 PM	Application	45 KB
inheritance.cpp	5/31/2022 8:48 PM	C++ Source File	3 KB
inheritance.exe	5/31/2022 8:48 PM	Application	48 KB
MaxNum.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
NameOfStudents.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
NameOfStudents.exe	5/31/2022 8:48 PM	Application	48 KB
new demo.txt	5/31/2022 9:44 PM	Text Document	1 KB
new.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
new1.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB

18 bytes

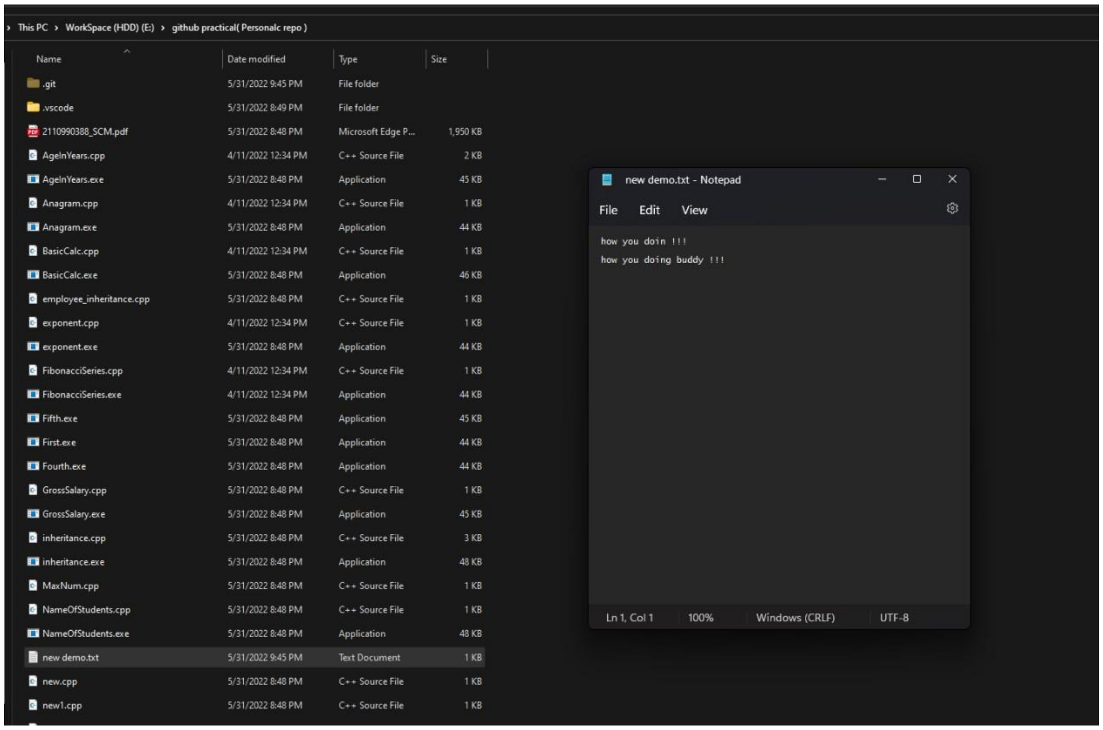
new demo.txt - Notepad

```
how you doin !!!
```

Step 3 – Switch to other branch that we already created called NewBranch by using the command ***git checkout (branch name)*** and added new text ‘ how you doing buddy !!! ’ in the new demo.txt file



```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git branch
  Experiment
  NewBranch
* master

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git checkout NewBranch
Switched to branch 'NewBranch'

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (NewBranch)
$ |
```



› This PC › WorkSpace (HDD) (E:) › github practical(Personalc repo)

Name	Date modified	Type	Size
.git	5/31/2022 9:45 PM	File folder	
.vscode	5/31/2022 8:49 PM	File folder	
2110990388_SCM.pdf	5/31/2022 8:48 PM	Microsoft Edge P...	1,950 KB
AgeInYears.cpp	4/11/2022 12:34 PM	C++ Source File	2 KB
AgeInYears.exe	5/31/2022 8:48 PM	Application	45 KB
Anagram.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
Anagram.exe	5/31/2022 8:48 PM	Application	44 KB
BasicCalc.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
BasicCalc.exe	5/31/2022 8:48 PM	Application	46 KB
employee_inheritance.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
exponent.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
exponent.exe	5/31/2022 8:48 PM	Application	44 KB
FibonacciSeries.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
FibonacciSeries.exe	4/11/2022 12:34 PM	Application	44 KB
Fifth.exe	5/31/2022 8:48 PM	Application	45 KB
First.exe	5/31/2022 8:48 PM	Application	44 KB
Fourth.exe	5/31/2022 8:48 PM	Application	44 KB
GrossSalary.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
GrossSalary.exe	5/31/2022 8:48 PM	Application	45 KB
inheritance.cpp	5/31/2022 8:48 PM	C++ Source File	3 KB
inheritance.exe	5/31/2022 8:48 PM	Application	48 KB
MaxNum.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
NameOfStudents.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
NameOfStudents.exe	5/31/2022 8:48 PM	Application	48 KB
new demo.txt	5/31/2022 9:45 PM	Text Document	1 KB
new.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
new1.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB

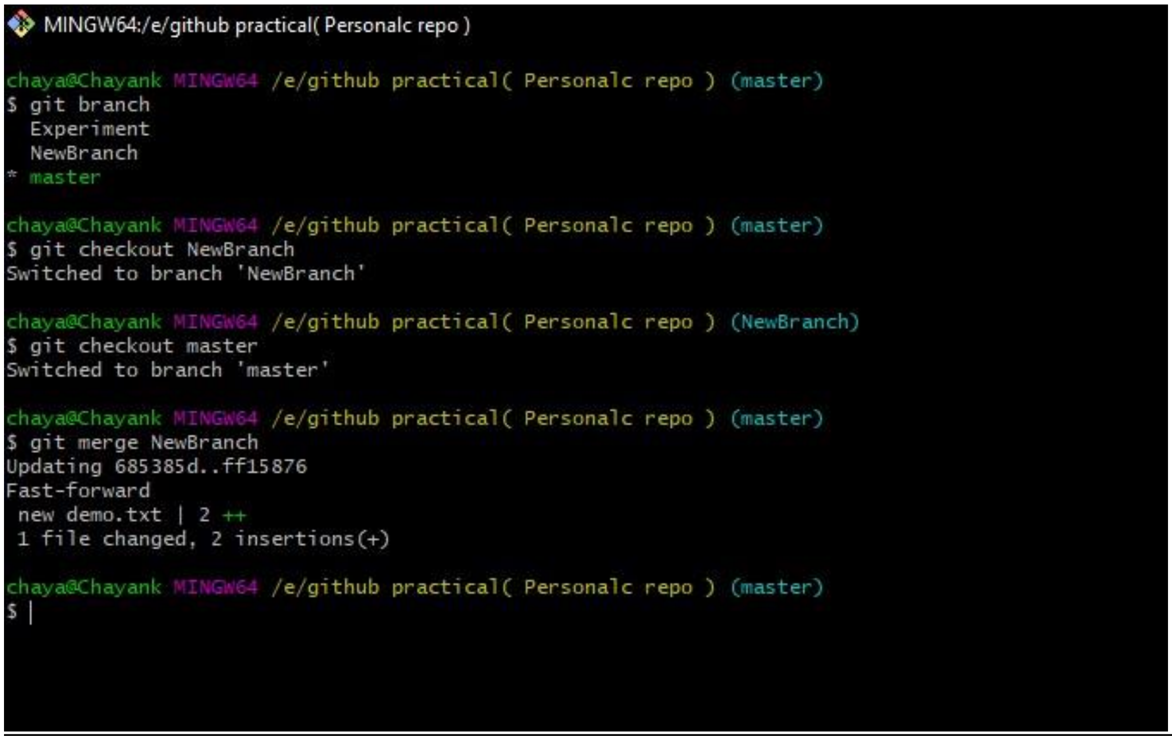
new demo.txt - Notepad

File Edit View

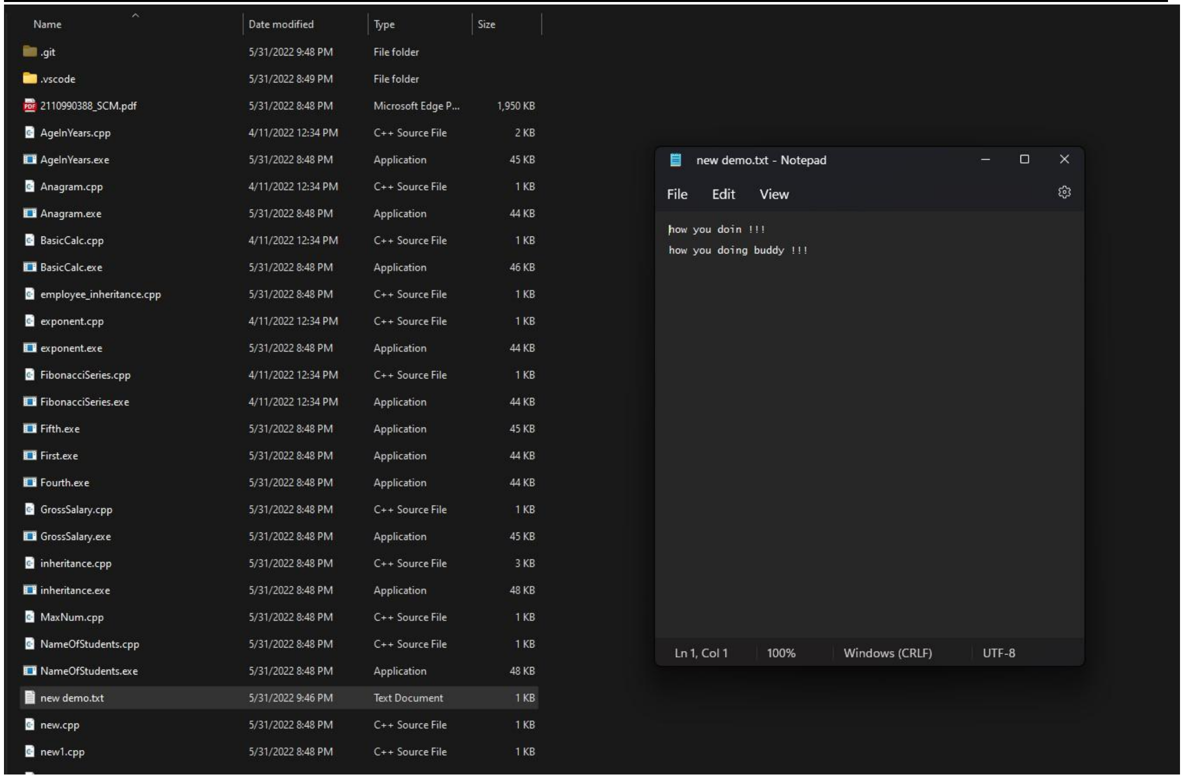
```
how you doin !!!
how you doing buddy !!!
```

Ln 1, Col 1 100% Windows (CRLF) UTF-8

Step 4 – Now switch to again master branch and merge NewBranch branch to the master branch, we can do that by using the command ***git merge NewBranch.*** Now we can see that in our master branch also the text appears that we only added in the branch NewBranch.



```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git branch
  Experiment
  NewBranch
* master

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git checkout NewBranch
Switched to branch 'NewBranch'

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (NewBranch)
$ git checkout master
Switched to branch 'master'

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git merge NewBranch
Updating 685385d..ff15876
Fast-forward
 new demo.txt | 2 ++
 1 file changed, 2 insertions(+)

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ |
```



Name	Date modified	Type	Size
.git	5/31/2022 9:48 PM	File folder	
.vscode	5/31/2022 8:49 PM	File folder	
2110990388_SCM.pdf	5/31/2022 8:48 PM	Microsoft Edge P...	1,950 KB
AgeInYears.cpp	4/11/2022 12:34 PM	C++ Source File	2 KB
AgeInYears.exe	5/31/2022 8:48 PM	Application	45 KB
Anagram.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
Anagram.exe	5/31/2022 8:48 PM	Application	44 KB
BasicCalc.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
BasicCalc.exe	5/31/2022 8:48 PM	Application	46 KB
employee_inheritance.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
exponent.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
exponent.exe	5/31/2022 8:48 PM	Application	44 KB
FibonacciSeries.cpp	4/11/2022 12:34 PM	C++ Source File	1 KB
FibonacciSeries.exe	4/11/2022 12:34 PM	Application	44 KB
Fifth.exe	5/31/2022 8:48 PM	Application	45 KB
First.exe	5/31/2022 8:48 PM	Application	44 KB
Fourth.exe	5/31/2022 8:48 PM	Application	44 KB
GrossSalary.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
GrossSalary.exe	5/31/2022 8:48 PM	Application	45 KB
inheritance.cpp	5/31/2022 8:48 PM	C++ Source File	3 KB
inheritance.exe	5/31/2022 8:48 PM	Application	48 KB
MaxNum.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
NameOfStudents.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
NameOfStudents.exe	5/31/2022 8:48 PM	Application	48 KB
new demo.txt	5/31/2022 9:46 PM	Text Document	1 KB
new.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB
new1.cpp	5/31/2022 8:48 PM	C++ Source File	1 KB

```
how you doin !!!
how you doing buddy !!!
```

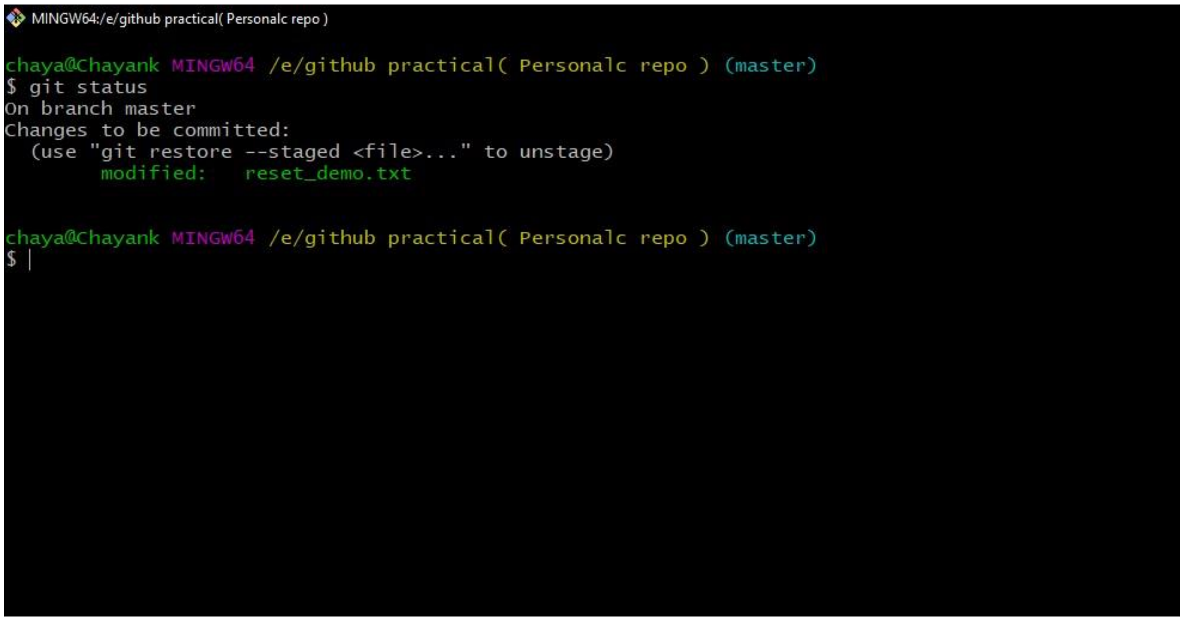
Experiment 9: Reset And Revert : git

reset :

git reset is used when we want to unstage a file and bring our changes back to the working directory. git reset can also be used to remove commits from the local repository.

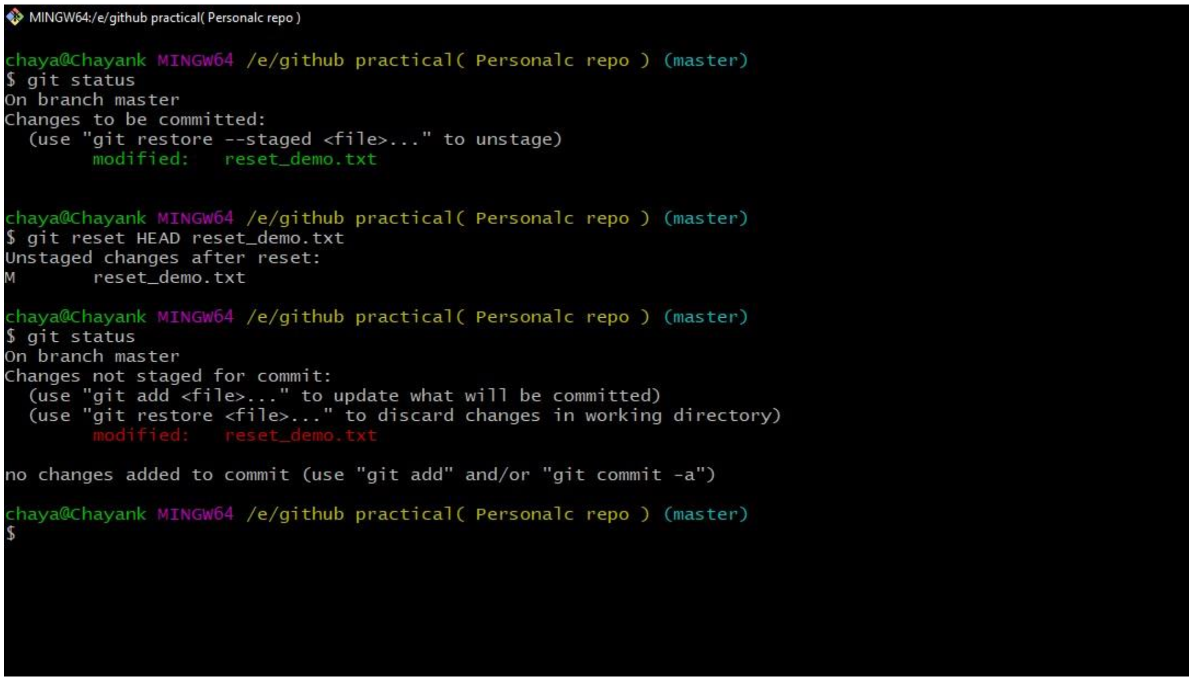
Following are the steps to reset :-

Step 1 -As we can see in the local repo we have a file name reset_demo.txt which is staged but not committed the changes.



```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   reset_demo.txt


chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ |
```

Step 2 – To unstaged the file we use the command ***git reset HEAD <filename>***

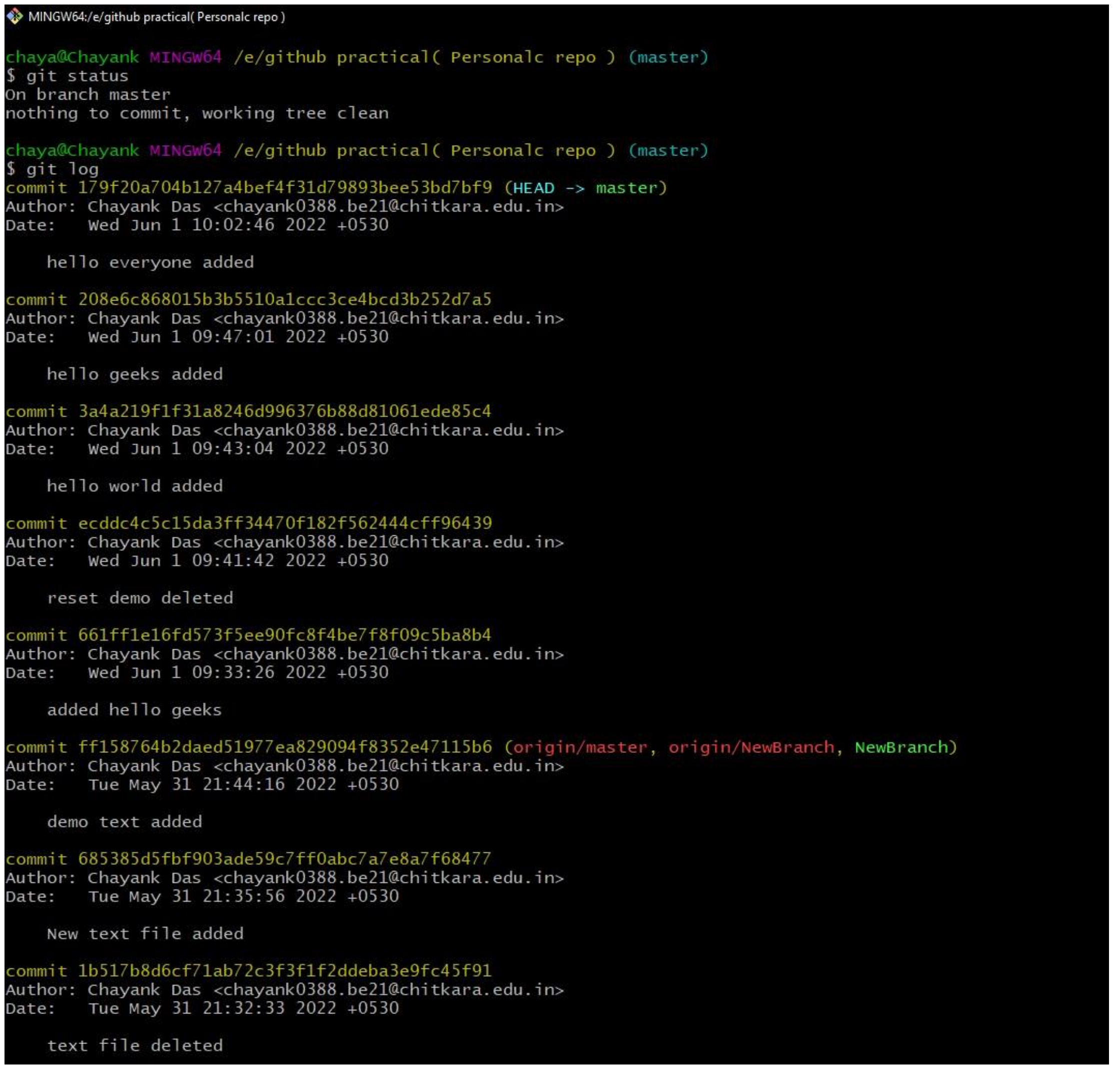


```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   reset_demo.txt


chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git reset HEAD reset_demo.txt
Unstaged changes after reset:
M       reset_demo.txt

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   reset_demo.txt

no changes added to commit (use "git add" and/or "git commit -a")

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$
```

We are back to the working directory, where our changes are present but the file is now unstaged. Now there are also some commits that we don't want to get committed and we want to remove them from our local repository. To see how to remove the commit from our local repository let's stage and commit the changes that we just did and then remove that commit.

Following are the steps to do the same:-

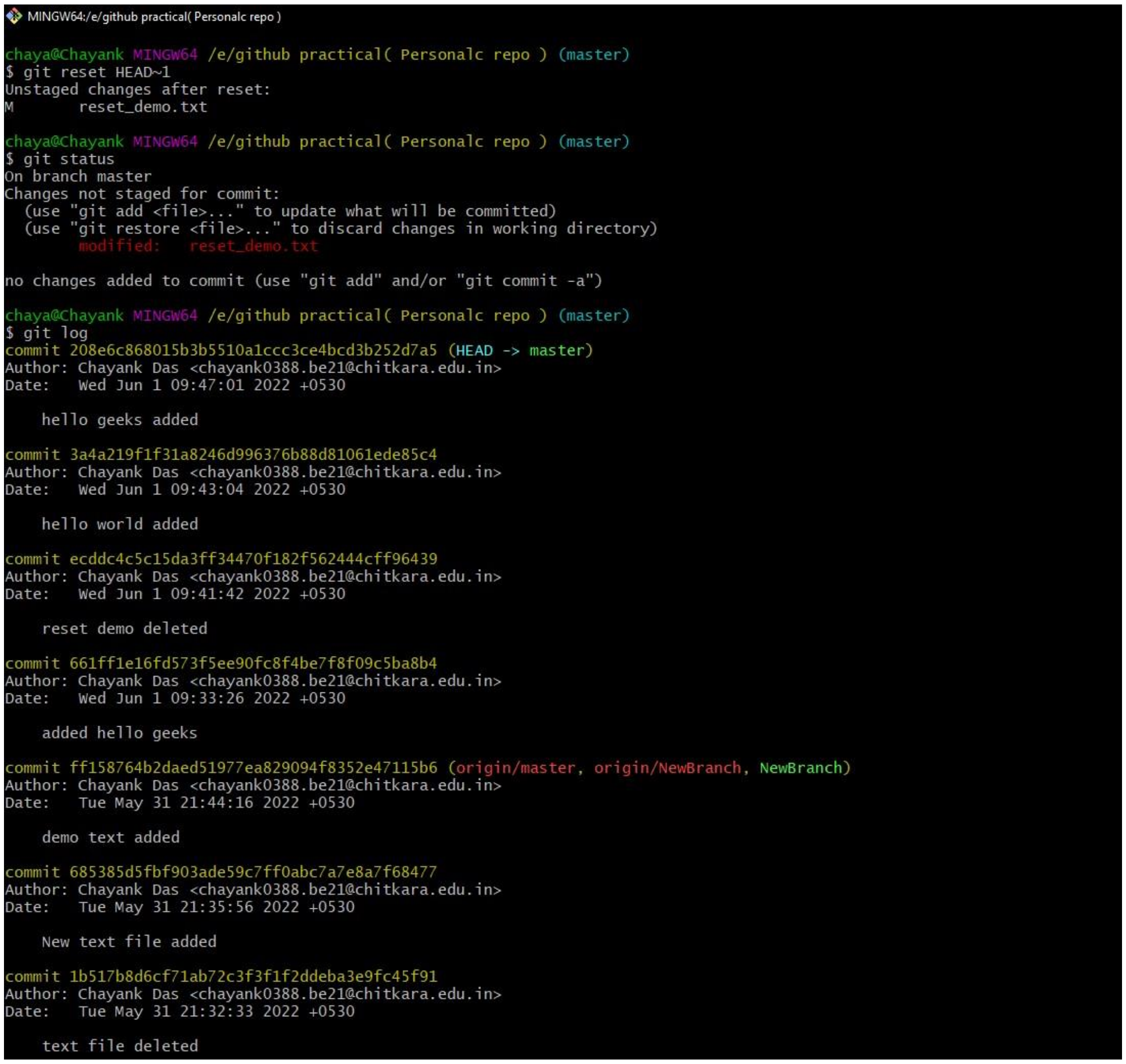
Step 1 – We have print out the all the commits using ***git log*** command, the latest commit is that we have an added new text "hello everyone" but we want to remove this commit-



```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git status
On branch master
nothing to commit, working tree clean

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git log
commit 179f20a704b127a4bef4f31d79893bee53bd7bf9 (HEAD -> master)
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 10:02:46 2022 +0530

    hello everyone added

commit 208e6c868015b3b5510a1ccc3ce4bcd3b252d7a5
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:47:01 2022 +0530

    hello geeks added

commit 3a4a219f1f31a8246d996376b88d81061ede85c4
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:43:04 2022 +0530

    hello world added

commit ecddc4c5c15da3ff34470f182f562444cff96439
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:41:42 2022 +0530

    reset demo deleted

commit 661ff1e16fd573f5ee90fc8f4be7f8f09c5ba8b4
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:33:26 2022 +0530

    added hello geeks

commit ff158764b2daed51977ea829094f8352e47115b6 (origin/master, origin/NewBranch, NewBranch)
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Tue May 31 21:44:16 2022 +0530

    demo text added

commit 685385d5fbf903ade59c7ff0abc7a7e8a7f68477
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Tue May 31 21:35:56 2022 +0530

    New text file added

commit 1b517b8d6cf71ab72c3f3f1f2ddeba3e9fc45f91
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Tue May 31 21:32:33 2022 +0530

    text file deleted
```

Step 2- To remove the latest commit we use this command ***git reset HEAD~1***.

After executing the command, we can see in the git log that the latest commit is deleted.



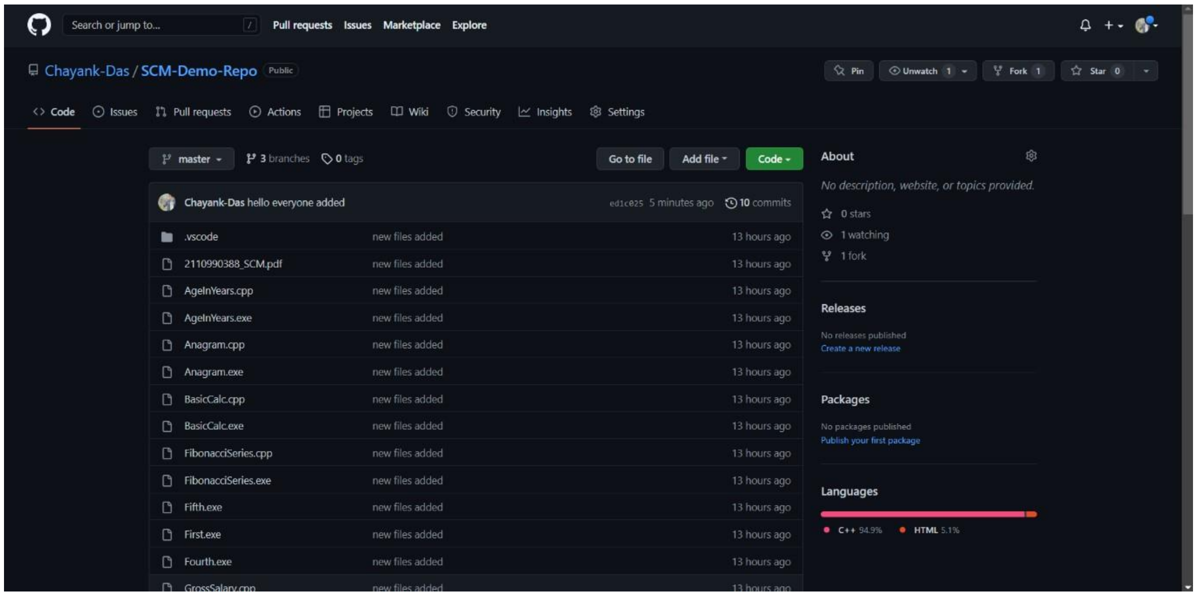
```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git reset HEAD~1
Unstaged changes after reset:
M       reset_demo.txt

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   reset_demo.txt

no changes added to commit (use "git add" and/or "git commit -a")

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git log
commit 208e6c868015b3b5510a1ccc3ce4bcd3b252d7a5 (HEAD -> master)
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:47:01 2022 +0530

    hello geeks added

commit 3a4a219f1f31a8246d996376b88d81061ede85c4
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:43:04 2022 +0530

    hello world added

commit ecddc4c5c15da3ff34470f182f562444cff96439
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:41:42 2022 +0530

    reset demo deleted

commit 661ff1e16fd573f5ee90fc8f4be7f8f09c5ba8b4
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:33:26 2022 +0530

    added hello geeks

commit ff158764b2daed51977ea829094f8352e47115b6 (origin/master, origin/NewBranch, NewBranch)
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Tue May 31 21:44:16 2022 +0530

    demo text added

commit 685385d5fbf903ade59c7ff0abc7a7e8a7f68477
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Tue May 31 21:35:56 2022 +0530

    New text file added

commit 1b517b8d6cf71ab72c3f3f1f2ddeba3e9fc45f91
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Tue May 31 21:32:33 2022 +0530

    text file deleted
```

git revert :

git revert is used to remove the commits from the remote repository. Since now our changes are in the working directory, let's add those changes to the staging area and commit them.

Following are the steps to do the same:-

Step 1- Now let's push our changes to the remote repository by using git push command.



Chayank-Das hello everyone added ed1c025 5 minutes ago 10 commits

.vscode	new files added	13 hours ago
2110990388_SCM.pdf	new files added	13 hours ago
AgeInYears.cpp	new files added	13 hours ago
AgeInYears.exe	new files added	13 hours ago
Anagram.cpp	new files added	13 hours ago
Anagram.exe	new files added	13 hours ago
BasicCalc.cpp	new files added	13 hours ago
BasicCalc.exe	new files added	13 hours ago
FibonacciSeries.cpp	new files added	13 hours ago
FibonacciSeries.exe	new files added	13 hours ago
Fifth.exe	new files added	13 hours ago
First.exe	new files added	13 hours ago
Fourth.exe	new files added	13 hours ago
GrossSalary.cpp	new files added	13 hours ago

No description, website, or topics provided.
0 stars
1 watching
1 fork

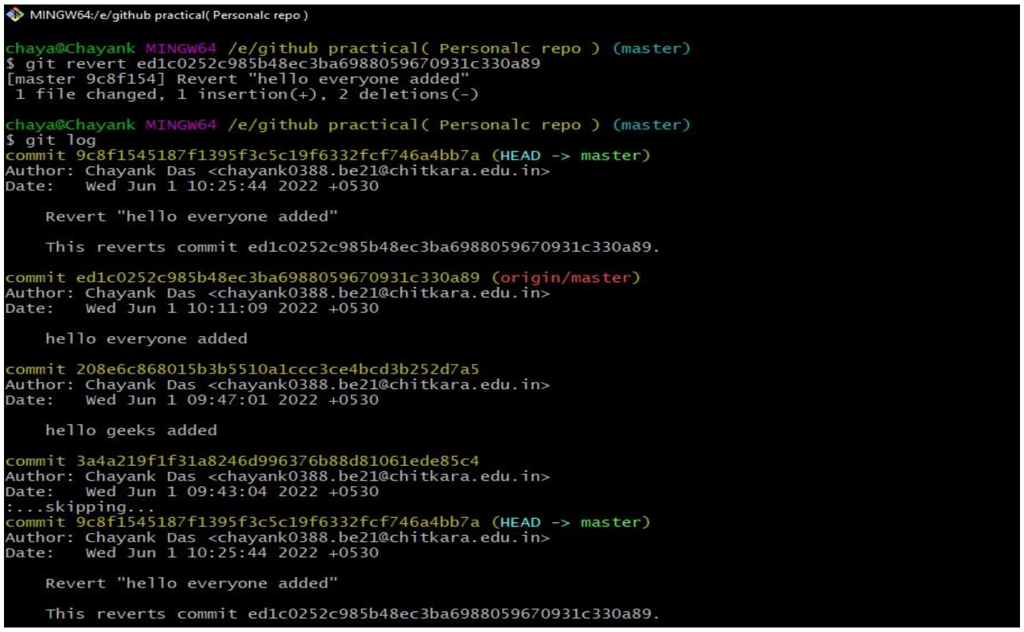
No releases published
Create a new release

No packages published
Publish your first package

● C++ 94.9% ● HTML 5.1%

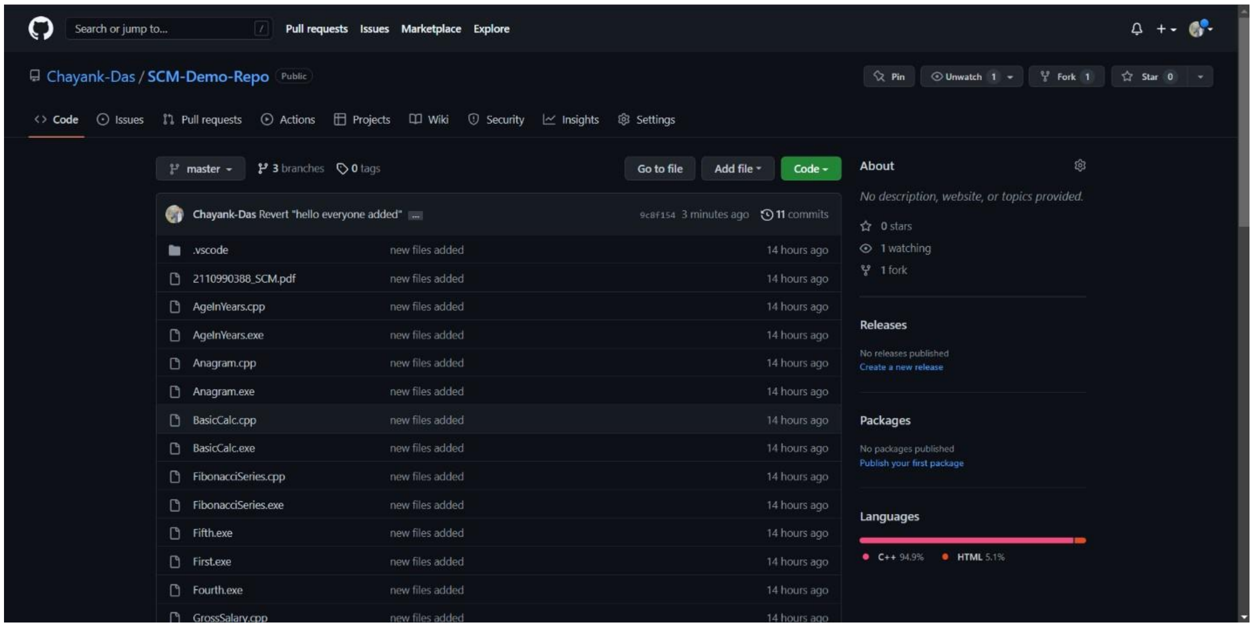
Now we want to delete the commit that we just added to the remote repository. We could have used the git reset command but that would have deleted the commit just from the local repository and not the remote repository. If we do this then we would get conflict that the remote commit is not present locally.

Step 2- git revert removes the commit that we have done but adds one more commit which tells us that the revert has been done. We can do this my using the command ***git revert <commit id>*****.** The git log shows that we have successfully reverted the commit.



```
MINGW64:/e/github practical( Personalc repo )

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git revert ed1c0252c985b48ec3ba6988059670931c330a89
[master 9c8f154] Revert "hello everyone added"
 1 file changed, 1 insertion(+), 2 deletions(-)

chaya@Chayank MINGW64 /e/github practical( Personalc repo ) (master)
$ git log
commit 9c8f1545187f1395f3c5c19f6332fcf746a4bb7a (HEAD -> master)
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 10:25:44 2022 +0530

    Revert "hello everyone added"

    This reverts commit ed1c0252c985b48ec3ba6988059670931c330a89.

commit ed1c0252c985b48ec3ba6988059670931c330a89 (origin/master)
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 10:11:09 2022 +0530

    hello everyone added

commit 208e6c868015b3b5510a1ccc3ce4bcd3b252d7a5
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:47:01 2022 +0530

    hello geeks added

commit 3a4a219f1f31a8246d996376b88d81061ede85c4
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 09:43:04 2022 +0530
:...skipping...
commit 9c8f1545187f1395f3c5c19f6332fcf746a4bb7a (HEAD -> master)
Author: Chayank Das <chayank0388.be21@chitkara.edu.in>
Date:   Wed Jun 1 10:25:44 2022 +0530

    Revert "hello everyone added"

    This reverts commit ed1c0252c985b48ec3ba6988059670931c330a89.
```

However since this commit is in local repository so we need to do ***git push*** so that our remote repository also notices that the change has been done.



Chayank-Das Revert "hello everyone added" 9c8f154 3 minutes ago 11 commits

.vscode	new files added	14 hours ago
2110990388_SCM.pdf	new files added	14 hours ago
AgeInYears.cpp	new files added	14 hours ago
AgeInYears.exe	new files added	14 hours ago
Anagram.cpp	new files added	14 hours ago
Anagram.exe	new files added	14 hours ago
BasicCalc.cpp	new files added	14 hours ago
BasicCalc.exe	new files added	14 hours ago
FibonacciSeries.cpp	new files added	14 hours ago
FibonacciSeries.exe	new files added	14 hours ago
Fifth.exe	new files added	14 hours ago
First.exe	new files added	14 hours ago
Fourth.exe	new files added	14 hours ago
GrossSalary.cpp	new files added	14 hours ago

About

No description, website, or topics provided.

0 stars
1 watching
1 fork

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

● C++ 94.9% ● HTML 5.1%

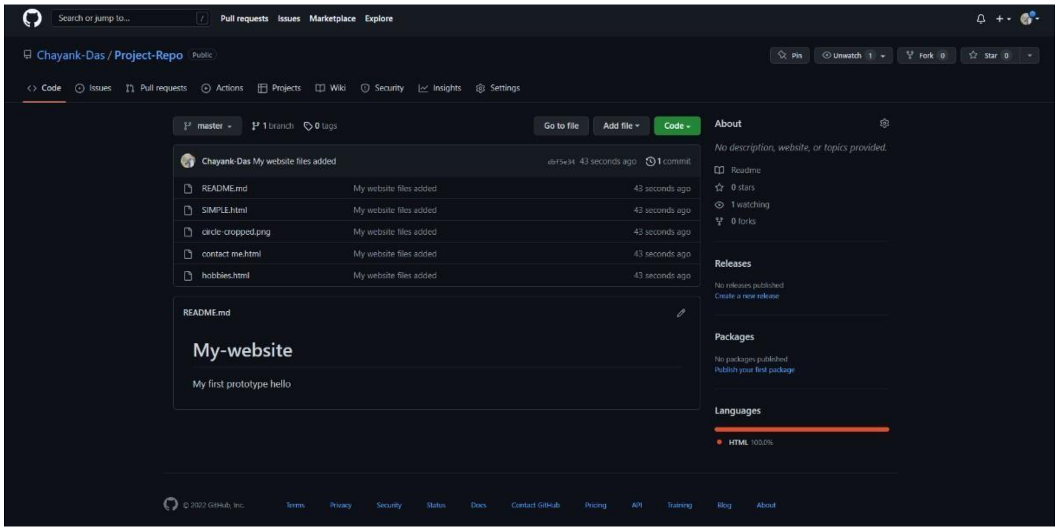
As we can see the revert command is also now reflected in the remote repo.

TASK 2

Experiment 1: Create a distributed Repository and add members in project team.

In simpler words, a repository on GitHub is a GitHub repository. Just like a repository on Git is a Git repository. A repository is similar to a folder and a GitHub repository becomes a folder that is available online on the cloud for the people to download, access, and contribute. This folder contains the code files of the project which can now be used by other people.

Step 1- Create and new repository or we can do it in our already made repository, and then click settings



Chayank-Das My website files added 43 seconds ago 1 commit

README.md	My website files added	43 seconds ago
SIMPLE.html	My website files added	43 seconds ago
circle-cropped.png	My website files added	43 seconds ago
contact me.html	My website files added	43 seconds ago
hobbies.html	My website files added	43 seconds ago

README.md

My-website

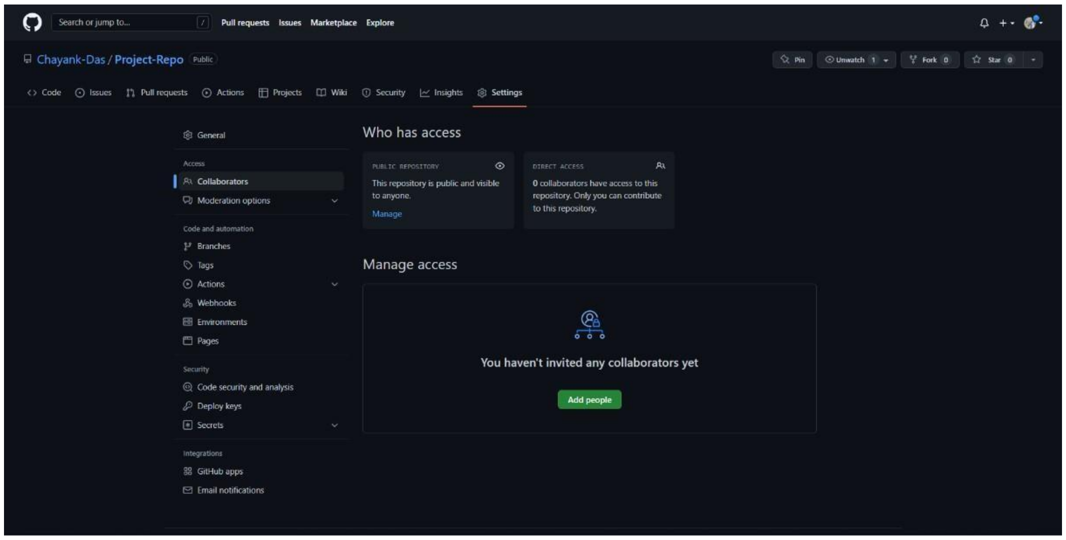
My first prototype hello

No description, website, or topics provided.

No releases published

No packages published

Step 2- Go to Collaborators option under the Settings tab. On the Collaborators page, you will see an Add people link as shown in the below diagram.



Who has access

PUBLIC REPOSITORY: This repository is public and visible to anyone. Manage

DIRECT ACCESS: 0 collaborators have access to this repository. Only you can contribute to this repository.

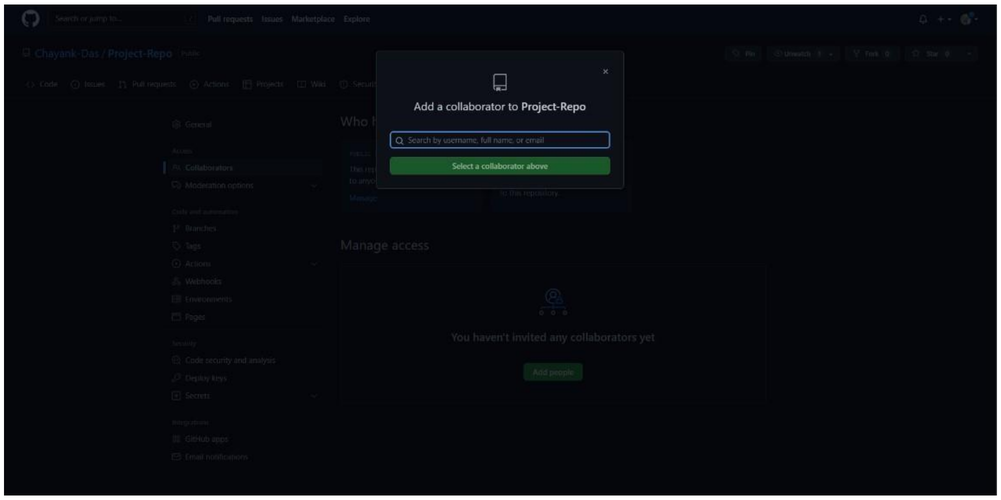
Manage access

You haven't invited any collaborators yet

Step 3 – You can Invite collaborators by any of the following options –

- Username
- Full name
- Email

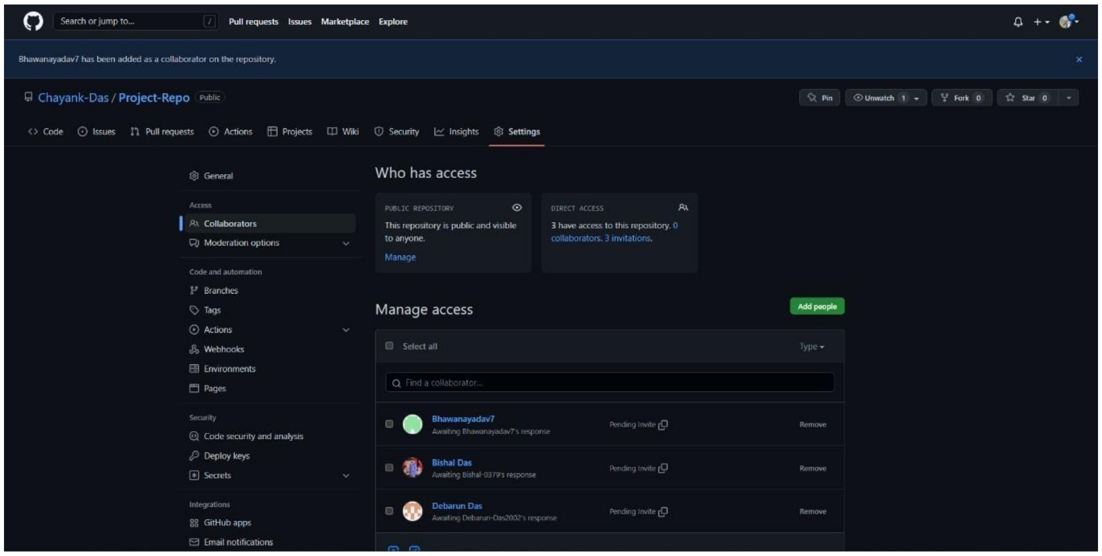
After you send the invite, the collaborator receives an email invitation. The collaborator has to accept it in order to get permission to collaborate on the same project.


Add a collaborator to Project-Repo

Search by username, full name, or email

Select a collaborator above
Manage access
You haven't invited any collaborators yet
Add people

The Manage Access option also allows a repository owner to view the invitations that are pending and not accepted.


Bhawanayadav7 has been added as a collaborator on the repository.

Chayank-Das / Project-Repo Public

Who has access

PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage

DIRECT ACCESS
3 have access to this repository. 0 collaborators. 3 invitations.

Manage access

Bhawanayadav7 Awaiting Bhawanayadav7's response	Pending Invite	Remove
Bishal Das Awaiting Bishal-0379's response	Pending Invite	Remove
Debarun Das Awaiting Debarun-Das2002's response	Pending Invite	Remove

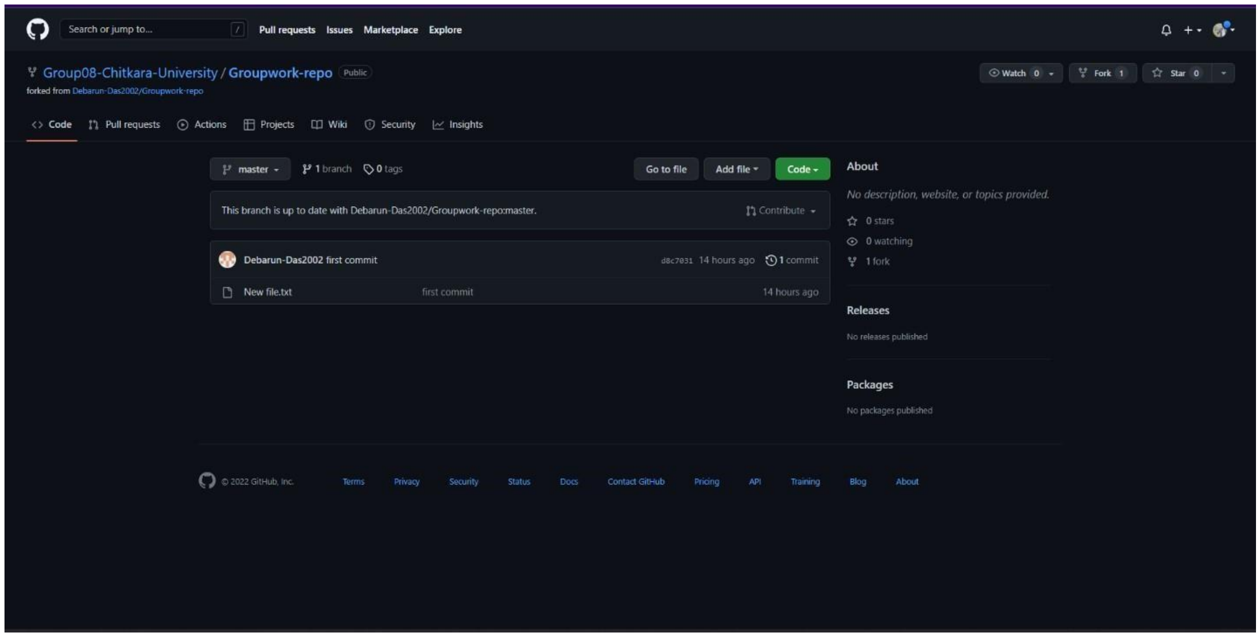
Experiment 2 & 3: Open and Close Pull Request and Each project member shall create a pull request on a team members repo and close pull requests generated by team members on own Repo as a maintainer.

What is a pull request?

Pull requests are the way we contribute to group projects or open source projects.

Steps to make a pull request –

Step 1- Fork the repository by clicking the fork button on the top of the page. This will create an instance of that entire repository in your account.



This branch is up to date with Debarun-Das2002/Groupwork-repo:master.

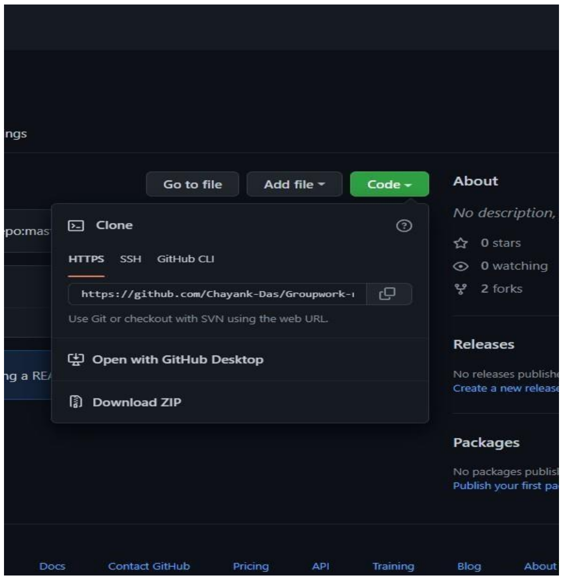
Debarun-Das2002 first commit d8c7031 14 hours ago 1 commit

New file.txt first commit 14 hours ago

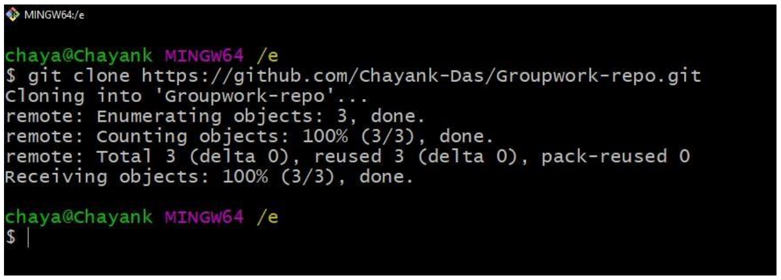
No description, website, or topics provided.

Step 2 - Once the repository is in your account, clone it to your machine to work with it locally.

To clone, click on the code button and copy the link.



Open the terminal and run the following command. It will clone the repository locally. git clone<copied link>



```
MINGW64:/e

chaya@Chayank MINGW64 /e
$ git clone https://github.com/Chayank-Das/Groupwork-repo.git
Cloning into 'Groupwork-repo'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.

chaya@Chayank MINGW64 /e
$
```

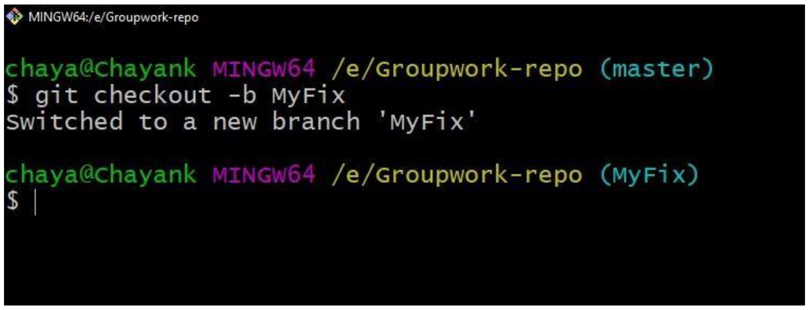
Now open the cloned folder and run git bash there

It’s good practice to create a new branch when working with repositories, whether it’s a small project or contributing to a group's work.

Branch name should be short and it should reflect the work we’re doing.

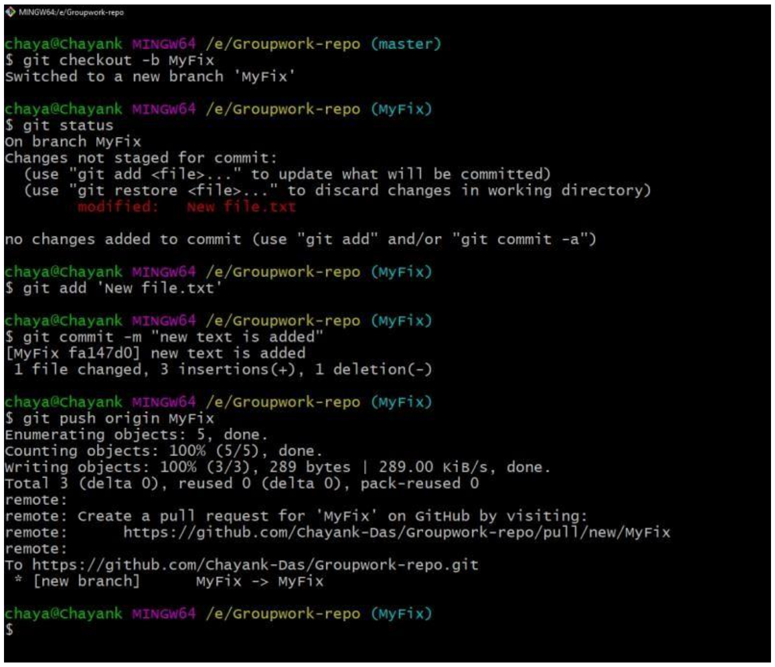
Step 3 -Now create a branch using the git checkout command:

$ git checkout -b [Branch Name]



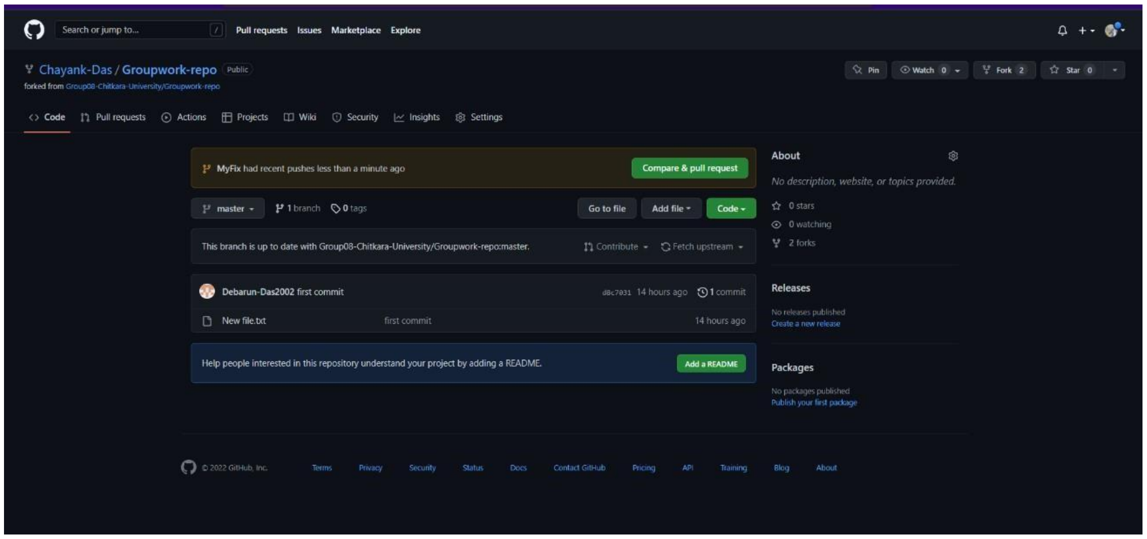
```
MINGW64:/e/Groupwork-repo

chaya@Chayank MINGW64 /e/Groupwork-repo (master)
$ git checkout -b MyFix
Switched to a new branch 'MyFix'

chaya@Chayank MINGW64 /e/Groupwork-repo (MyFix)
$ |
```

Step 4 – Make changes in the files and then staged it, commit the changes and then push it to the remote repo

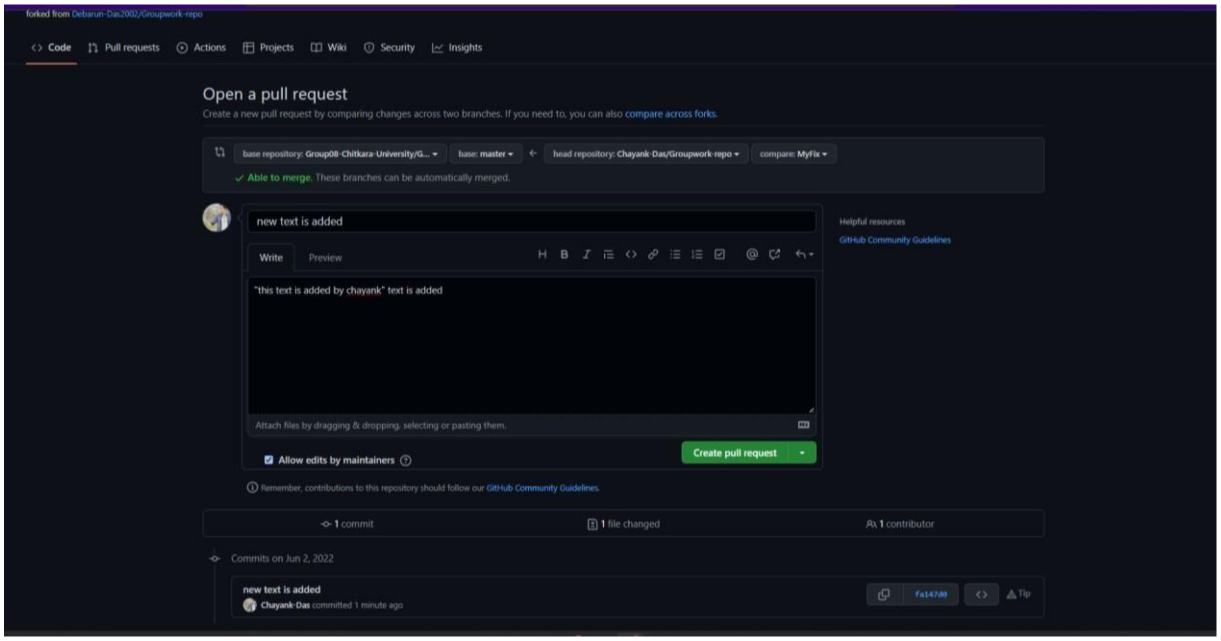


```
MINGW64:/e/Groupwork-repo

chaya@Chayank MINGW64 /e/Groupwork-repo (master)
$ git checkout -b MyFix
Switched to a new branch 'MyFix'

chaya@Chayank MINGW64 /e/Groupwork-repo (MyFix)
$ git status
On branch MyFix
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   New file.txt

no changes added to commit (use "git add" and/or "git commit -a")

chaya@Chayank MINGW64 /e/Groupwork-repo (MyFix)
$ git add 'New file.txt'

chaya@Chayank MINGW64 /e/Groupwork-repo (MyFix)
$ git commit -m "new text is added"
[MyFix fa147d0] new text is added
 1 file changed, 3 insertions(+), 1 deletion(-)

chaya@Chayank MINGW64 /e/Groupwork-repo (MyFix)
$ git push origin MyFix
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 289 bytes | 289.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'MyFix' on GitHub by visiting:
remote:      https://github.com/Chayank-Das/Groupwork-repo/pull/new/MyFix
remote:
To https://github.com/Chayank-Das/Groupwork-repo.git
 * [new branch]      MyFix -> MyFix

chaya@Chayank MINGW64 /e/Groupwork-repo (MyFix)
$
```

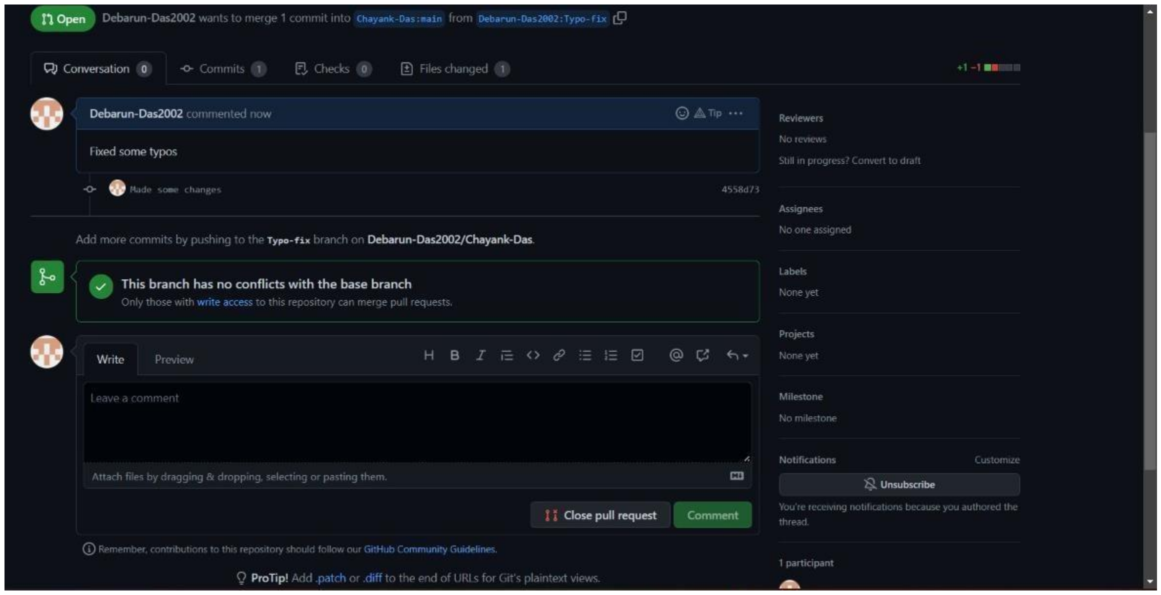
Step 5- Now we need to create the pull request to the original repo or in the master branch. We can see in our forked repository that a button appears saying "Compare & pull request**".**



Clicking the button will give you a page where we can write the description of what changes have been made and finally the button to create the pull request.


forked from Debarun-Das2002/Groupwork-repo
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
Open a pull request
Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.
base repository: Group08-Chitkara-University/G...
base: master
head repository: Chayank-Das/Groupwork-repo
compare: MyFix
Able to merge. These branches can be automatically merged.
new text is added
Write
Preview
"this text is added by chayank" text is added
Attach files by dragging & dropping, selecting or pasting them.
Allow edits by maintainers
Create pull request
Remember, contributions to this repository should follow our GitHub Community Guidelines.
Helpful resources
GitHub Community Guidelines
1 commit
1 file changed
1 contributor
Commits on Jun 2, 2022
new text is added
Chayank-Das committed 1 minute ago
fa147d0

Now the person will receive a pull request in his/her repository. Clicking the pull request button, the person can see what changes are made and whether the person will merge the changes if necessary or he/she can close the pull request if the changes are not necessary.


Open Debarun-Das2002 wants to merge 1 commit into Chayank-Das:main from Debarun-Das2002:Typo-fix

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 −1

Debarun-Das2002 commented now

Fixed some typos

Made some changes 4558d73

Add more commits by pushing to the Typo-fix branch on Debarun-Das2002/Chayank-Das.

This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Close pull request Comment

Remember, contributions to this repository should follow our GitHub Community Guidelines.

ProTip! Add .patch or .diff to the end of URLs for Git's plaintext views.

Reviewers
No reviews
Still in progress? Convert to draft

Assignees
No one assigned

Labels
None yet

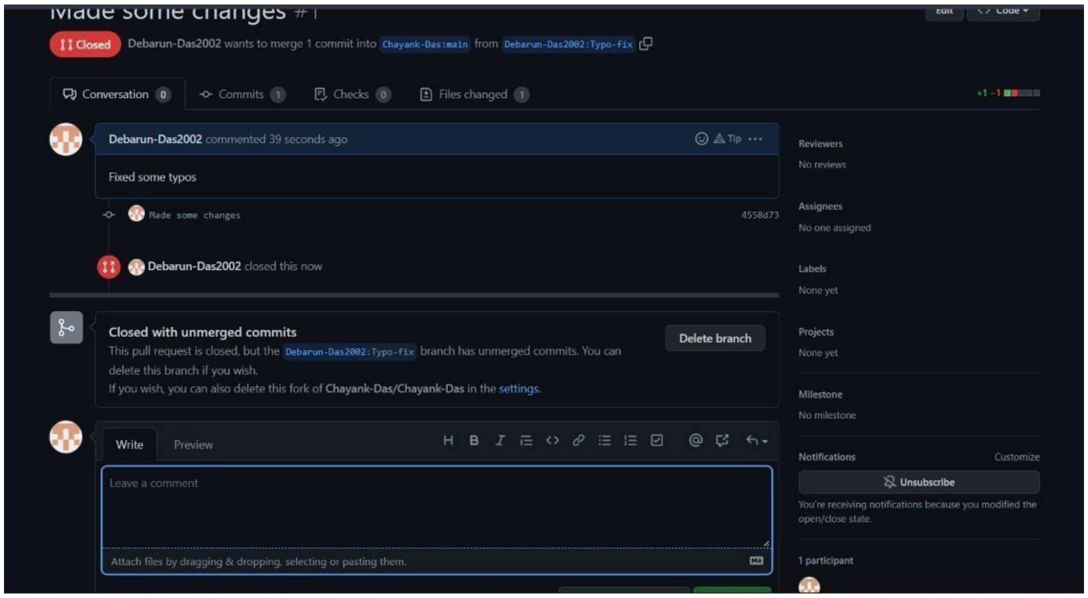
Projects
None yet

Milestone
No milestone

Notifications Customize
Unsubscribe
You're receiving notifications because you authored the thread.

1 participant

Here for instance the owner closes the pull request as he/she may not need the changes that we have made.


Made some changes #1

Closed Debarun-Das2002 wants to merge 1 commit into Chayank-Das:main from Debarun-Das2002:Typo-fix

Conversation 0 Commits 1 Checks 0 Files changed 1 +1 −1

Debarun-Das2002 commented 39 seconds ago

Fixed some typos

Made some changes 4558d73

Debarun-Das2002 closed this now

Closed with unmerged commits
This pull request is closed, but the Debarun-Das2002:Typo-fix branch has unmerged commits. You can delete this branch if you wish.
If you wish, you can also delete this fork of Chayank-Das/Chayank-Das in the settings.

Delete branch

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Reviewers
No reviews

Assignees
No one assigned

Labels
None yet

Projects
None yet

Milestone
No milestone

Notifications Customize
Unsubscribe
You're receiving notifications because you modified the open/close state.

1 participant

Experiment 3: Publish and Print network graphs

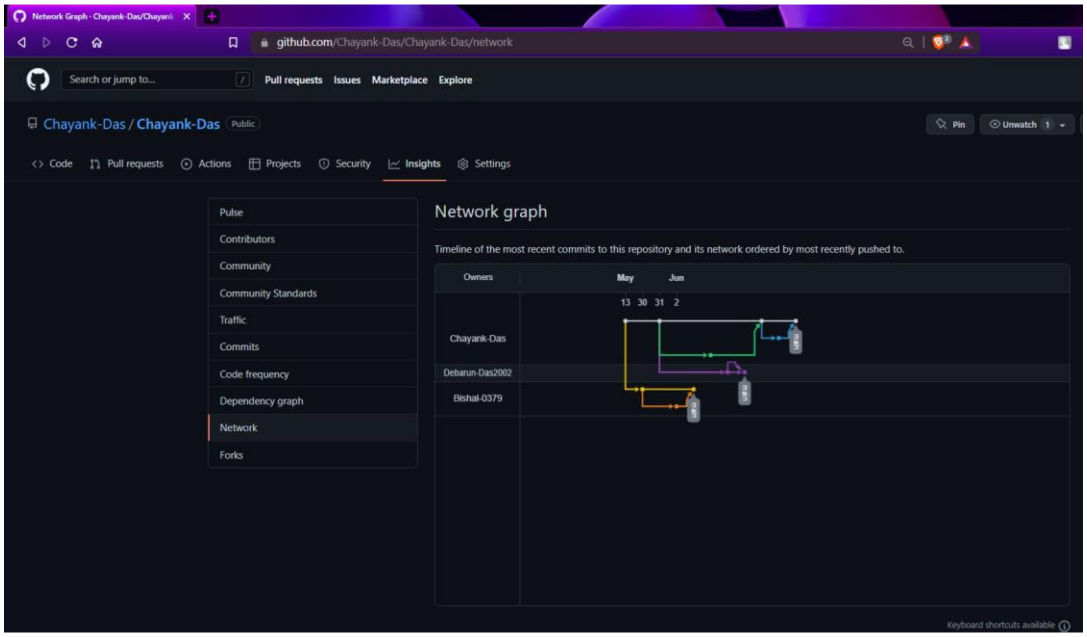
The network graph displays the branch history of the entire repository network, including branches of the root repository and branches of the forks that contains commits unique to the network.

Steps to see the network graphs –

Step 1- On the remote repository, navigate to the main page of the repository.

Step 2- Under your repository name, click Insights.

Step 3- In the left sidebar, click Network.



Timeline of the most recent commits to this repository and its network ordered by most recently pushed to.