

**File
of
Source Code Management**

**Name: Bhawana Yadav
Roll No: 2110990367
Group: 8
Cluster: Beta**

Index

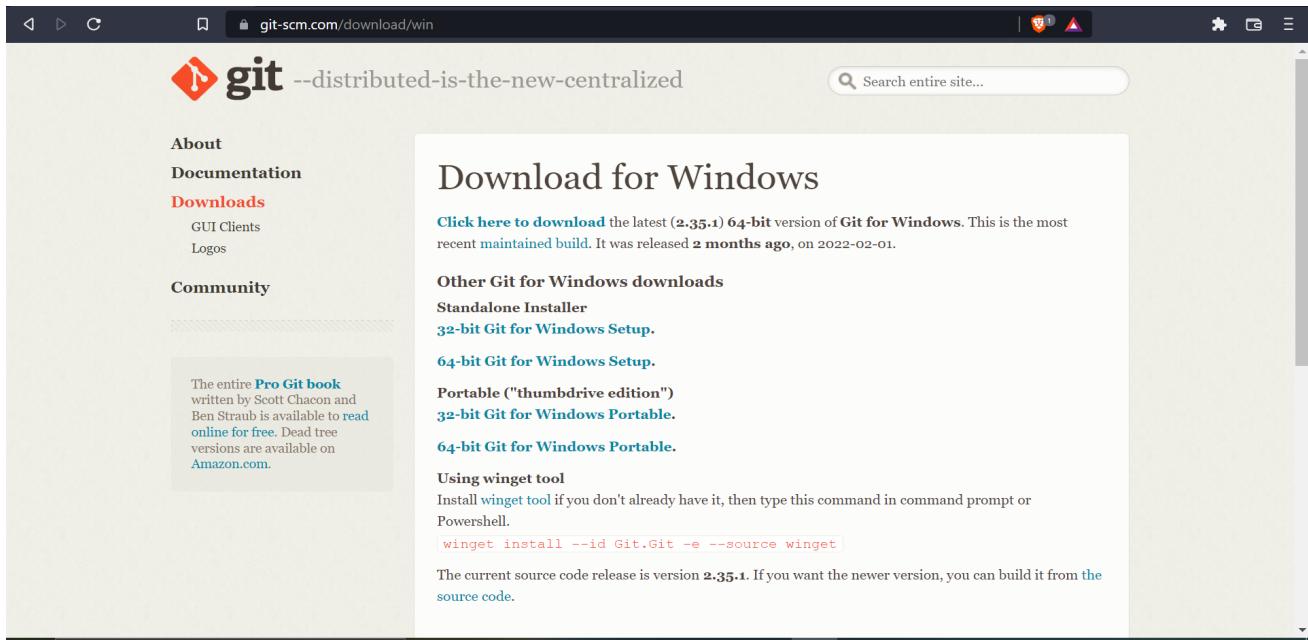
Task	Experiment	Topic	Page No.
Task 1.1	1	Setting up Git Client	3-6
	2	setting up GitHub Account	7-8
	3	How to use Git log	9-12
	4	Create and visualize Branch	13-14
	5	Life Cycle of the Git	15-20
Task 1.2	6	Add collaborators on repo	21-23
	7	Fork and commit	24-25
	8	Merge and resolve conflicts	26-36
	9	Reset and Revert	37-40
Task 2	10	Project	41-50
	11	Network Graph	51

Experiment 1

Aim: Setting up Git **client**.

Description: Git installation: Git client can be installed on various devices(Windows,Mac or Linux) from <https://git-scm.com/download/win>

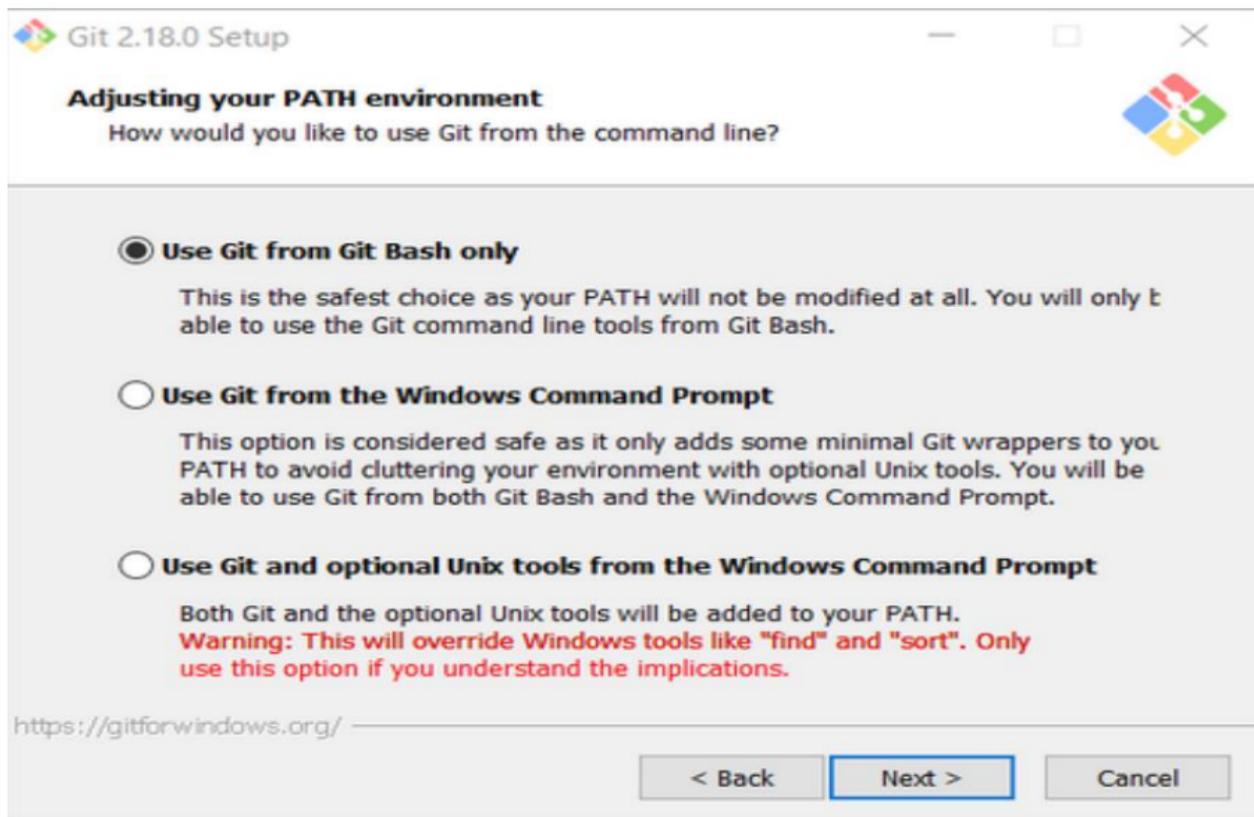
You can always go for the default settings. It is absolutely fine or as per your choices you can change the default settings while installation.



The screenshot shows the official Git website at git-scm.com/download/win. The page title is "git --distributed-is-the-new-centralized". On the left, there's a sidebar with links for "About", "Documentation", "Downloads" (which includes "GUI Clients" and "Logos"), and "Community". A sidebar box mentions the "Pro Git book" by Scott Chacon and Ben Straub. The main content area is titled "Download for Windows" and instructs users to click here to download the latest 64-bit version. It also lists other download options like "Standalone Installer", "32-bit Git for Windows Setup", and "Portable" versions. A section on using "winget tool" is shown with a command example: `winget install --id Git.Git -e --source winget`. A note at the bottom states the current source code release is version 2.35.1.

I will recommend going for default settings.
In the Adjusting your PATH screen, all three options are acceptable:

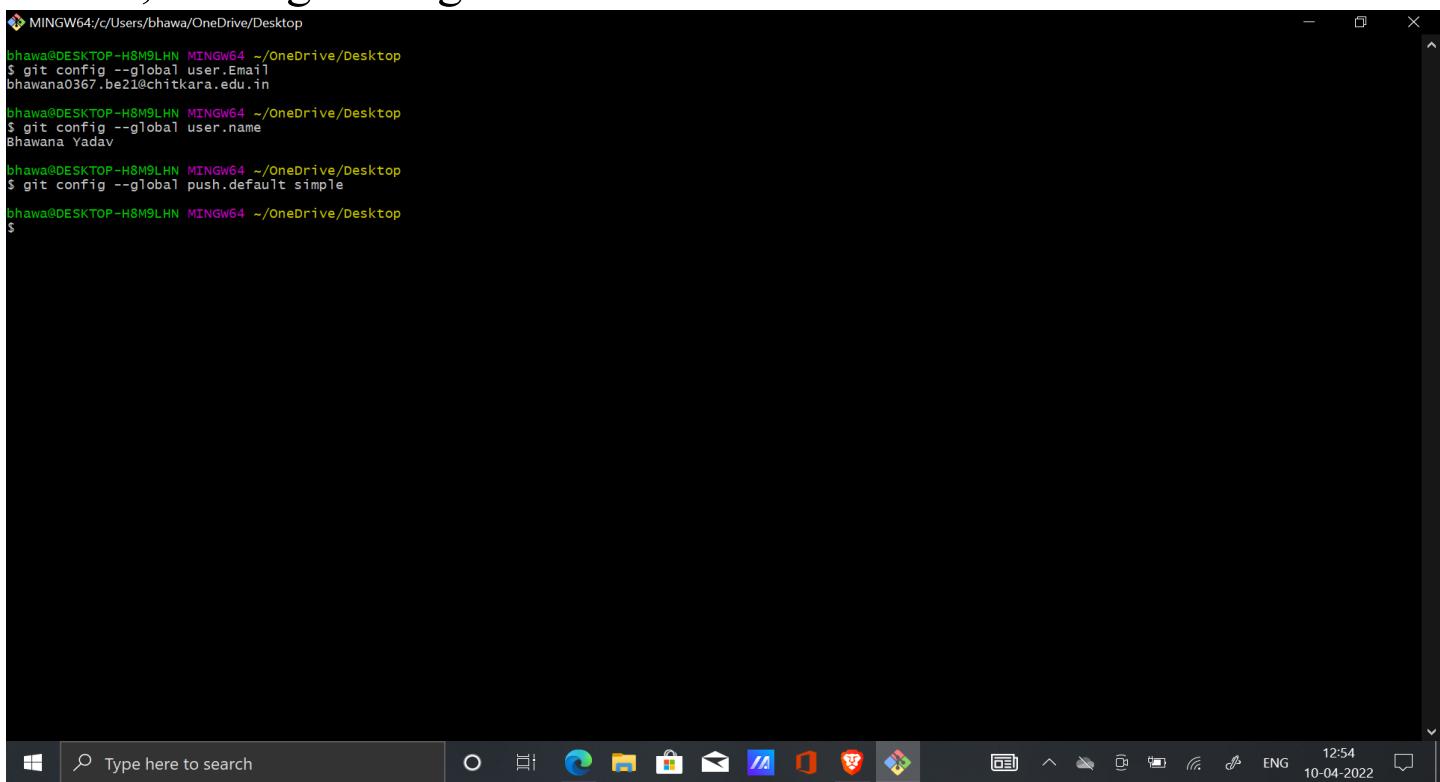
1. Use Git from Git Bash only: no integration, and no extra command in your command path.
2. Use Git from the windows Command Prompt: add flexibility – you can simply run git from a windows command prompt, and is often the setting for people in industry – but this does add some extra commands.
3. Use Git and optional Unix tools from the Windows Command Prompt: this is also a robust choice and useful if you like to use Unix like commands like grep.



After the installation process is complete, there is some custom configuration you must do. Follow the instructions below

- a. From within File Explorer, right-click on any folder. A context menu appears containing the commands "Git Bash here" and "GitGUI here". These commands permit you to launch either Git client. For now, select Git Bash here.
- b. Enter the command `git config --global user.Email "name@msoe.edu"` This links your Git activity to your email address. Without this, your commits will often show up as "unknown login". Replace name with your own MSOE email name.

- c. Enter the command `git config --global user.name "Your Name"` Git uses this to log your activity. Replace "Your Name" by your actual first and last name.
- d. Enter the command `git config --global push.default simple` This ensures that all pushes go back to the branch from which they were pulled. Otherwise pushes will go to the master branch, forcing a merge.



The screenshot shows a Windows terminal window titled 'MINGW64/c/Users/bhawa/OneDrive/Desktop'. It contains the following command-line session:

```
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ git config --global user.Email
bhawana0367.be21@chitkara.edu.in
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ git config --global user.name
bhawana Yadav
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ git config --global push.default simple
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$
```

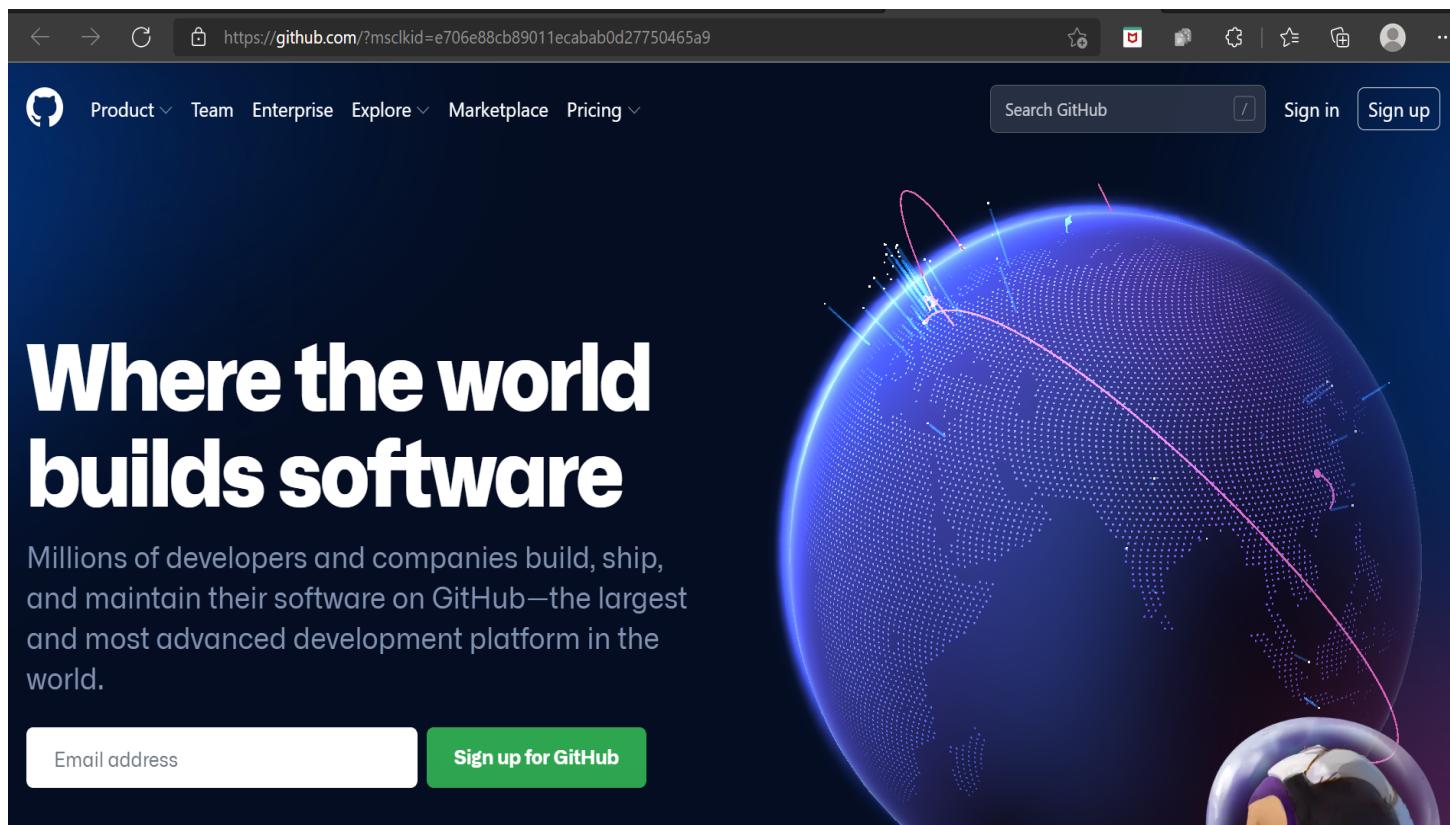
The taskbar at the bottom of the screen includes icons for File Explorer, Edge, File Manager, Mail, Microsoft Store, and others, along with system status indicators like battery level and signal strength. The date and time '10-04-2022 12:54' are also visible.

Experiment 2

Aim: Setting up **GitHub Account**

Description:

Step 1. Creating an account: To sign up for an account on GitHub.com, navigate to [GitHub: Where the world builds software · GitHub](#) and follow the prompts. To keep your GitHub account secure you should use a strong and unique password. For more information, see “Creating a strong password”.



Step 2. Choosing your GitHub product: You can choose GitHub Free or GitHub Pro to get access to different features for your personal account. You can upgrade at any time if you are unsure at first which product you want.

Step 3. Verifying your email address: To ensure you can use all the features in your GitHub plan, verify your email address after signing up for a new account.

Step 4. Different Options: there will be different options to select from you can select according to your need and set up your GitHub account successfully.

Experiment 3

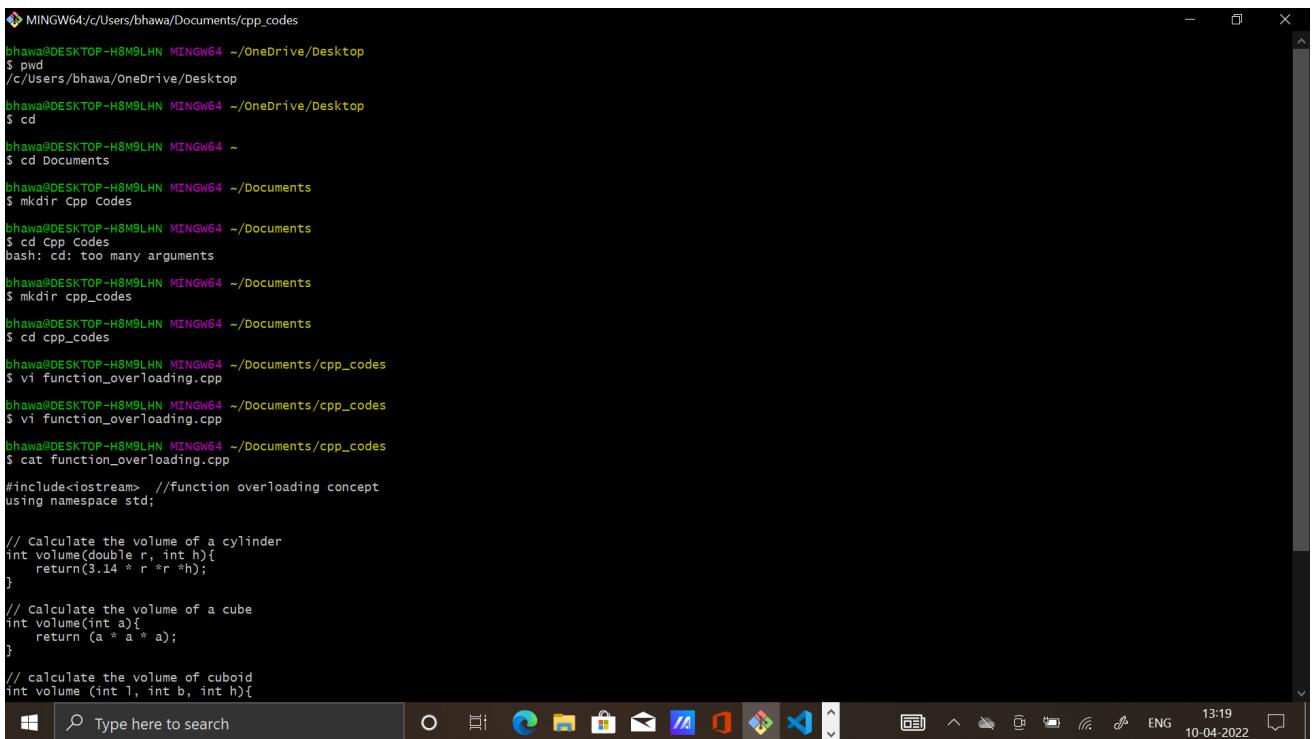
Aim: How to use Git log
learning basic Git commands and how to make a repository.

Description: We Will take a look at some basic commands

1. ls: To list all the content of the current working directory.
2. Clear: To clear the page
3. pwd: Tells about the current working directory
4. cd: changes the directory

5. mkdir: to create new directory
6. vi: Editor that can be used during creating code files for
eg:vi hello.cpp
7. :wq: write and quit used for saving the file
8. cat: to see the content inside a file

All these commands are shown in the screenshots below you can also take help from there

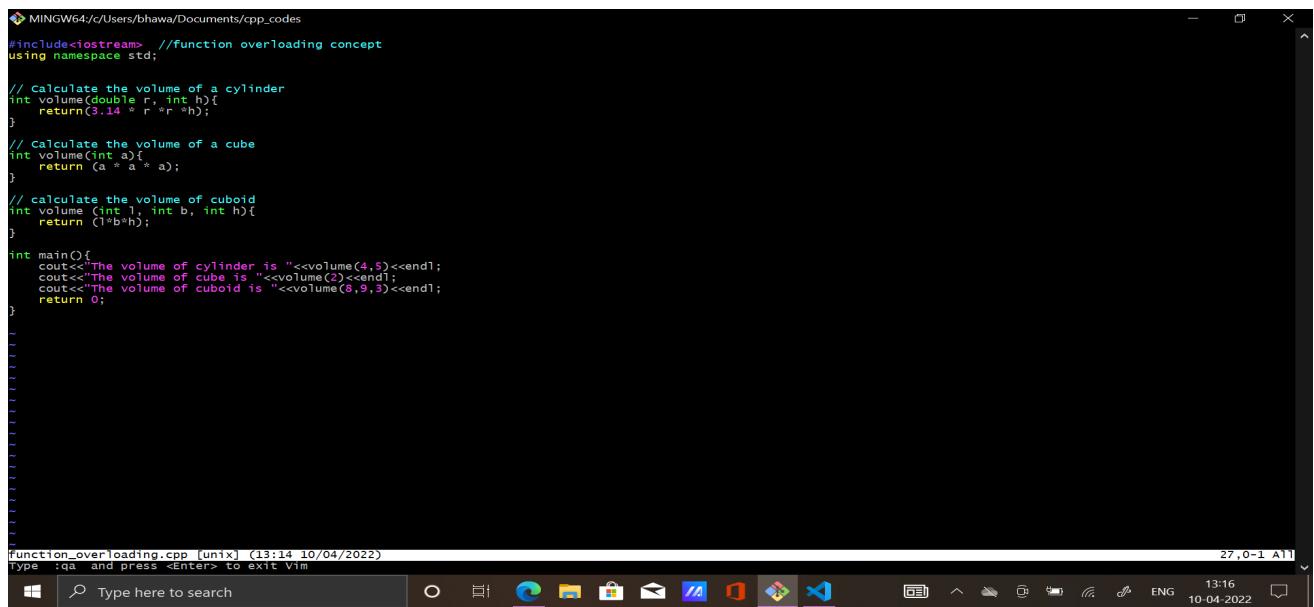


```
MINGW64:/c/Users/bhawa/Documents/cpp_codes
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ pwd
/c/Users/bhawa/OneDrive/Desktop
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ cd
bhawa@DESKTOP-H8M9LHN MINGW64 ~
$ cd Documents
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents
$ mkdir Cpp_Codes
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents
$ cd Cpp_Codes
bash: cd: too many arguments
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents
$ mkdir cpp_codes
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents
$ cd cpp_codes
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes
$ vi function_overloading.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes
$ vi function_overloading.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes
$ cat function_overloading.cpp
#include<iostream> //function overloading concept
using namespace std;

// calculate the volume of a cylinder
int volume(double r, int h){
    return(3.14 * r *r *h);
}

// calculate the volume of a cube
int volume(int a){
    return (a * a * a);
}

// calculate the volume of cuboid
int volume (int l, int b, int h){
```



```

MINGW64:/c/Users/bhawa/Documents/cpp_codes
#include<iostream> //function overloading concept
using namespace std;

// Calculate the volume of a cylinder
int volume(double r, int h){
    return(3.14 * r * r * h);
}

// Calculate the volume of a cube
int volume(int a){
    return (a * a * a);
}

// calculate the volume of cuboid
int volume (int l, int b, int h){
    return (l*b*h);
}

int main(){
    cout<<"The volume of cylinder is "<<volume(4,5)<<endl;
    cout<<"The volume of cube is "<<volume(2)<<endl;
    cout<<"The volume of cuboid is "<<volume(8,9,3)<<endl;
    return 0;
}

function_overloading.cpp [unix] (13:14 10/04/2022)
Type :q! and press <Enter> to exit vim

```

Running Git commands and making a repository

1. git status: Tells about the status of a repository.
2. git init: converts the folder into Repository.
3. git add: adds file.
4. git commit -m "message": to commit inside the Repository.
5. git log: shows commit status.
6. git remote and origin "url link": refers to remote repository and origin is the default remote name in git
7. git push -u origin master: to push the files into repository

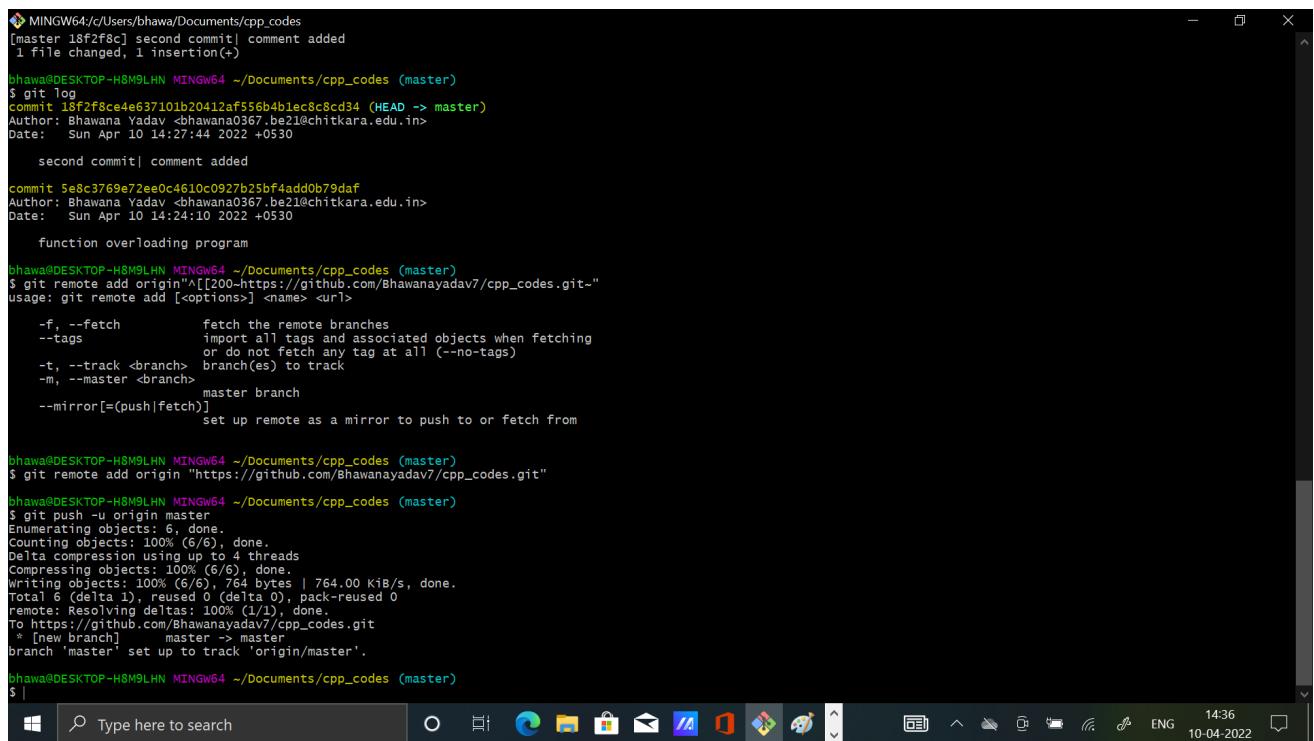
```
MINGW64:/c/Users/bhawa/Documents/cpp_codes
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes
$ pwd
/c/Users/bhawa/Documents/cpp_codes
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes
$ git status
fatal: not a git repository (or any of the parent directories): .git
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes
$ git init
Initialized empty Git repository in C:/Users/bhawa/Documents/cpp_codes/.git/
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ ls
function_overloading.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ ls -ah
./ ./ .git/ function_overloading.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git status
On branch master
No commits yet
Untracked files:
 (use "git add <file>..." to include in what will be committed)
   function_overloading.cpp
nothing added to commit but untracked files present (use "git add" to track)
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git add .
warning: LF will be replaced by CRLF in function_overloading.cpp.
The file will have its original line endings in your working directory
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git status
On branch master
No commits yet
Changes to be committed:
 (use "git rm --cached <file>..." to unstage)
  new file:   function_overloading.cpp
```

```
MINGW64:/c/Users/bhawa/Documents/cpp_codes
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git commit -m"function overloading program"
[master (root-commit) 5e8c376] function overloading program
 1 file changed, 27 insertions(+)
 create mode 100644 function_overloading.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git status
On branch master
nothing to commit, working tree clean
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git log
commit 5e8c3769e72ee0c4610c0927b25bf4add0b79daf (HEAD -> master)
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Sun Apr 10 14:24:10 2022 +0530

    function overloading program
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ vi function_overloading.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git log
commit 5e8c3769e72ee0c4610c0927b25bf4add0b79daf (HEAD -> master)
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Sun Apr 10 14:24:10 2022 +0530

    function overloading program
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (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:   function_overloading.cpp

no changes added to commit (use "git add" and/or "git commit -a")
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git add .
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
```



```

MINGW64:/c/Users/bhawa/Documents/cpp_codes
[master 18faf8c] second commit| comment added
 1 file changed, 1 insertion(+)

bhawa@DESKTOP-H5M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git log
commit 18f2f8ce4e637101b20412af556b4b1ec8c8cd34 (HEAD -> master)
Author: Bhawana Yadav <bhawanaya0367.bez1@chitkara.edu.in>
Date:   Sun Apr 10 14:27:44 2022 +0530

    second commit| comment added

commit 5e8c3769e72ae0c4610c0927b25bf4add0b79daf
Author: Bhawana Yadav <bhawanaya0367.bez1@chitkara.edu.in>
Date:   Sun Apr 10 14:24:10 2022 +0530

  function overloading program

bhawa@DESKTOP-H5M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git remote add origin "[200-https://github.com/Bhawanayadav7/cpp_codes.git]"
usage: git remote add [<options>] <name> <url>

-f, --fetch          fetch the remote branches
--tags              import all tags and associated objects when fetching
                   or do not fetch any tag at all (--no-tags)
-t, --track <branch> branch(es) to track
-m, --master <branch>
                   master branch
--mirror[=(push|fetch)] set up remote as a mirror to push to or fetch from

bhawa@DESKTOP-H5M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git push -u origin master
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 764 bytes | 764.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/Bhawanayadav7/cpp_codes.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.

bhawa@DESKTOP-H5M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ |

```

These pictures can help you in setting up a repository and in committing and pushing files.

Experiment 4

Aim: Create and visualize branches in Git

~ What are branches?

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master. As you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, the master branch pointer moves forward automatically.

~ How to create branches?

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

~What happens when you create a new branch?

Well, doing so creates a new pointer for you to move around. Let's say you want to create a new branch called testing. You do this with the git branch command: \$ git branch testing

```
→ git git:(master) git checkout alpha
Switched to branch 'alpha'
→ git git:(alpha) git branch
* alpha
  master
→ git git:(alpha) git checkout -b beta
Switched to a new branch 'beta'
→ git git:(beta) git branch
  alpha
* beta
  master
→ git git:(beta) git checkout alpha
Switched to branch 'alpha'
→ git git:(alpha) touch style.css
→ git git:(alpha) X ||
```

Experiment 5

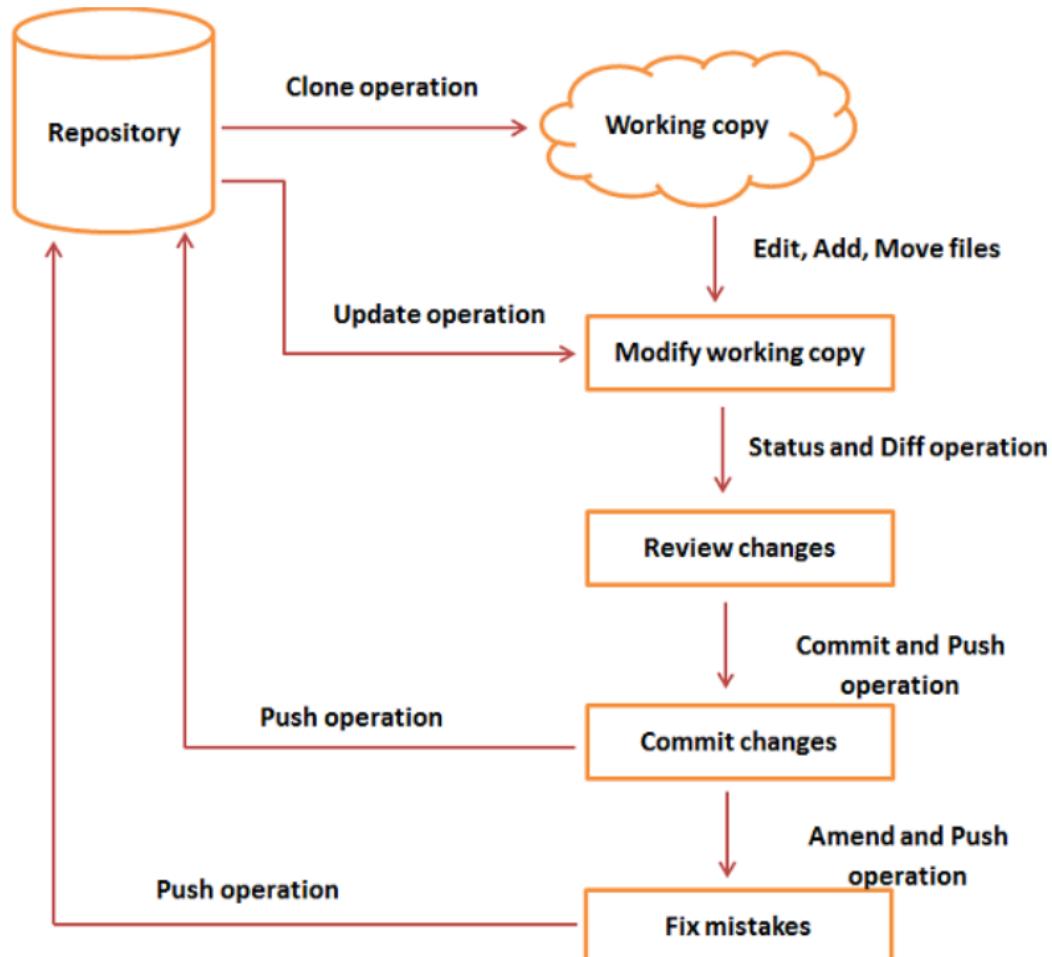
Aim: Git lifecycle description

General workflow is as follows –

- You clone the Git repository as a working copy.
- You modify the working copy by adding/editing files.
- If necessary, you also update the working copy by taking other developer's changes.
- You review the changes before commit.
- You commit changes. If everything is fine, then you push the changes to the repository.
- After committing, if you realize something is wrong, then you correct the last commit and push the changes to the repository.

Work-flow

Shown below is the pictorial representation of the work-flow.





```

MINGW64:/c/Users/bhawa/OneDrive/Desktop/Cpp_Programs
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ git clone "https://github.com/Bishal-0379"
Cloning into 'Bishal-0379'...
remote: Not Found
fatal: repository 'https://github.com/Bishal-0379/' not found

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ git clone "https://github.com/Bishal-0379/Cpp_Programs.git"
Cloning into 'Cpp_Programs'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 41 (delta 4), reused 41 (delta 4), pack-reused 0
Receiving objects: 100% (41/41), 503.35 KiB | 690.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ ls
Cpp_Programs/ 'Microsoft Edge.lnk'* 'Zoom.lnk*' desktop.ini

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs
$ cd Cpp_Programs
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ git log --oneline
579c635 (HEAD -> main, origin/main, origin/HEAD) first commit

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ cat Cpp_Programs
cat: Cpp_Programs: No such file or directory

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ ls -ah
./           12_sum_digits.cpp    17_percentagegrade.cpp  21_oddnumberarray.cpp  5_vaccine.cpp      for_loop.cpp     swap_notthirdvar
..          12_sum_digits.exe*  17_percentagegrade.exe*  22_2darray.cpp       6_grading.cpp    input_keyboard.cpp swap_thirdvari
.git/        13_seriscalculate.cpp 18_checkdigit.cpp   22_part2.cpp       7_alphabet.cpp  post_pre_inc_dec.cpp switch_oper.cp
.vscode/    14_notes.cpp       19_sumarray.cpp    2_absolutevalue.cpp  8_weekday.cpp   programs.docx    tempCodeRunner
10_largestno.cpp 15_triangle.cpp 1_checkpositive.cpp  3_shopdiscount.cpp  9_multiple3,7.cpp  programs.pdf    test.cpp
11_factorial.cpp 16_triangletype.cpp 20_largeelement.cpp 4_gradingsystem.cpp array.cpp       size_datatypes.cpp

```



```

MINGW64:/c/Users/bhawa/OneDrive/Desktop/Cpp_Programs
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ cat for_loop.cpp
#include <iostream>
using namespace std;

int main()
{
    for (int i = 1; i <= 5; i++)
    {
        cout << "Hello World\n";
    }
    return 0;
}

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ for_loop.cpp
bash: for_loop.cpp: command not found

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ vi for_loop.cpp

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

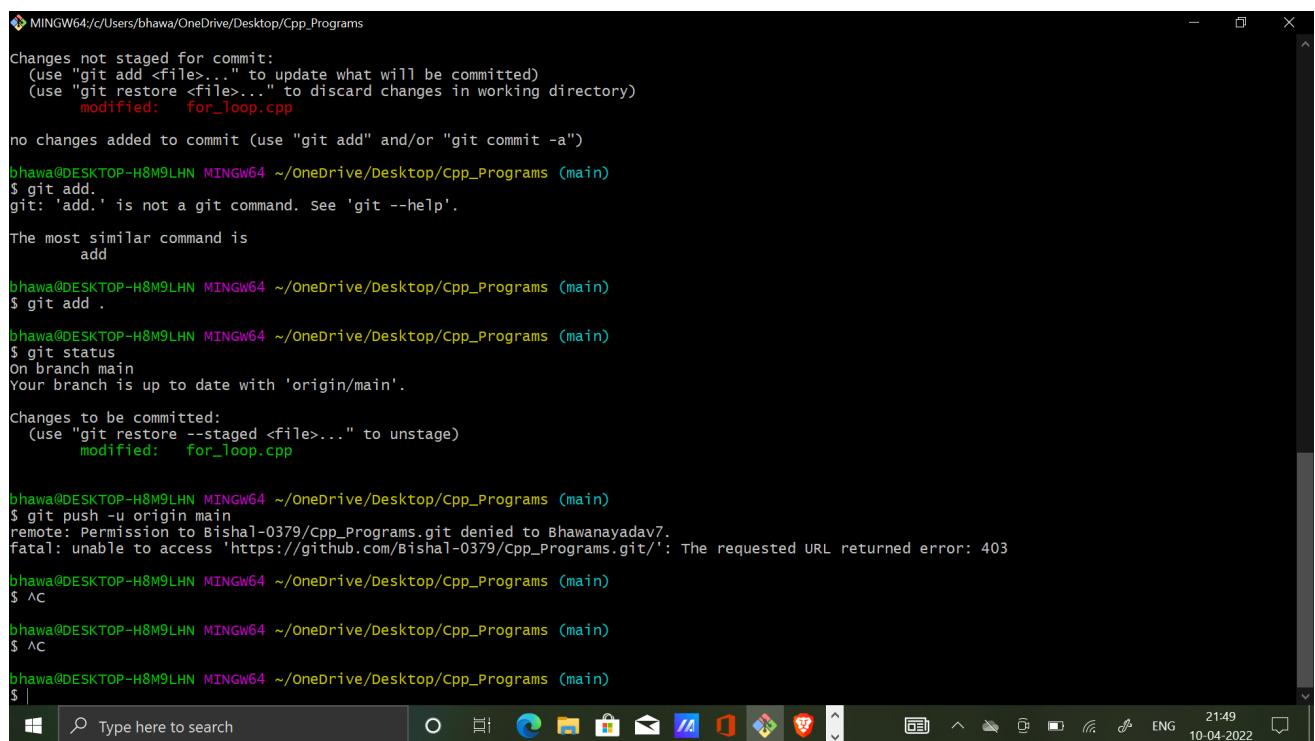
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   for_loop.cpp

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

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
  add


```



```
MINGW64/c/Users/bhawa/OneDrive/Desktop/Cpp_Programs

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:   for_loop.cpp

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

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ git add.
git: 'add.' is not a git command. See 'git --help'.
The most similar command is
  add

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   for_loop.cpp

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ git push -u origin main
remote: Permission to Bishal-0379/cpp_Programs.git denied to Bhawanayadav7.
fatal: unable to access 'https://github.com/Bishal-0379/Cpp_Programs.git/': The requested URL returned error: 403

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ ^C

bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/Cpp_Programs (main)
$ |
```

so it showed permission denied which is because we did not forked it and tried to make changes in some else repository which is not allowed.

so let us learn to fork in git client

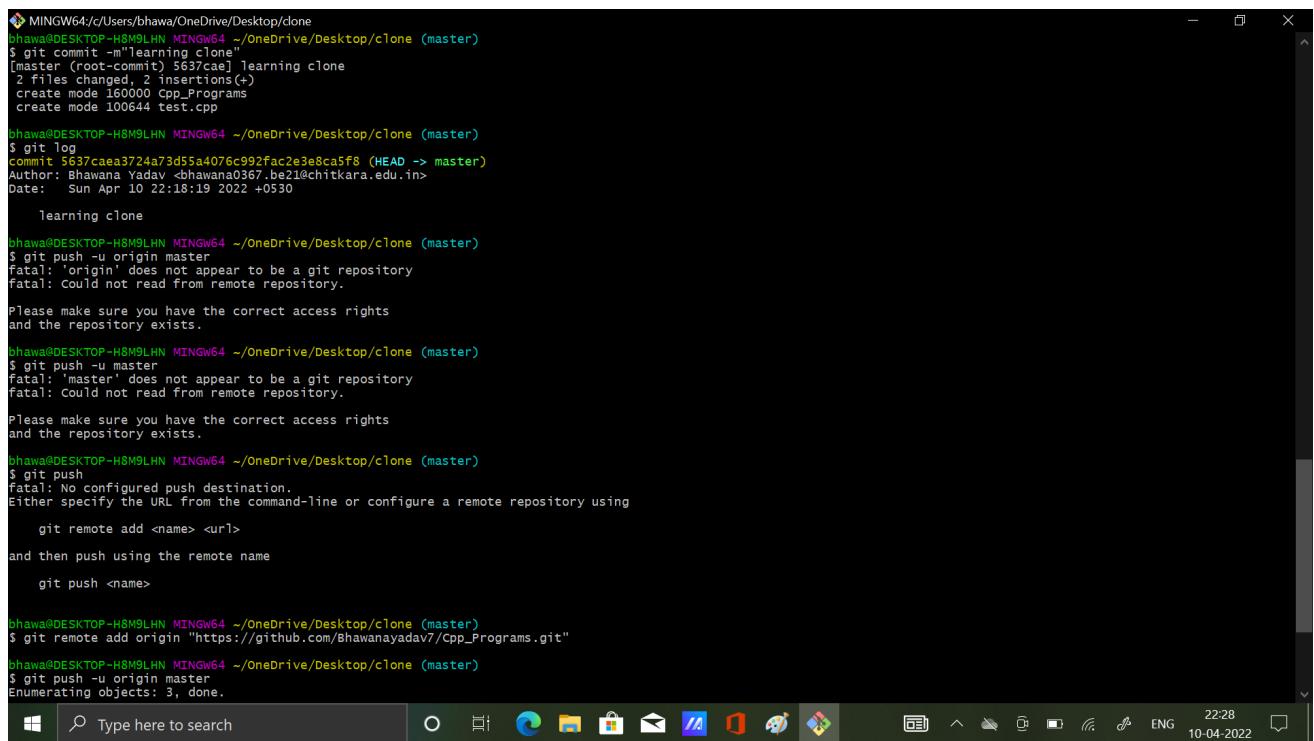


```
MINGW64:/c/Users/bhawa/OneDrive/Desktop/clone
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ cd clone
bash: cd: No such file or directory
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ mkdir clone
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop
$ cd clone
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone
$ git clone "https://github.com/Bhawanayadav7/Cpp_Programs.git"
Cloning into 'Cpp_Programs'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (35/35), done.
remote: Total 41 (delta 4), reused 41 (delta 4), pack-reused 0
Receiving objects: 100% (41/41), 503.35 KiB | 459.00 KiB/s, done.
Resolving deltas: 100% (4/4), done.
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone
$ git log --oneline
fatal: not a git repository (or any of the parent directories): .git
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone
$ git init
Initialized empty Git repository in C:/Users/bhawa/OneDrive/Desktop/clone/.git/
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git log --oneline
fatal: your current branch 'master' does not have any commits yet
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ ls -ah
./ .../.git/ Cpp_Programs/
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ ls Cpp_Programs
10_largestno.cpp      15_triangle.cpp      1_checkpositive.cpp    3_shopdiscount.cpp   9_multiple3,7.cpp      programs.pdf          test.cpp
11_factorial.cpp      16_triangletype.cpp  20_largеlement.cpp    4_gradingssystem.cpp  array.cpp           size_datatypes.cpp
12_sum_digits.cpp     17_percentagegrade.cpp 21_однодиизмерныйarray.cpp  5_vaccine.cpp       forLoop.cpp        swap_notthirdvariable.cpp
12_sum_digits.exe*   17_percentagegrade.exe* 22_2darray.cpp       6_grading.cpp       input_keyboard.cpp  swap_thirdvariable.cpp
13_seriescalculator.cpp 18_checkdigit.cpp   22_part2.cpp       7_alphabet.cpp     post_pre_inc_dec.cpp switch_oper.cpp
14_notes.cpp          19_Sumarray.cpp    2_absolutevalue.cpp  8_weekday.cpp      programs.docx       tempCodeRunnerFile.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ cat for_loop.cpp
cat: for_loop.cpp: No such file or directory

```

```
MINGW64:/c/Users/bhawa/OneDrive/Desktop/clone
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ vi test.cpp
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git status
On branch master
No commits yet
Untracked files:
  (use "git add <file>" to include in what will be committed)
    Cpp_Programs/
      test.cpp
nothing added to commit but untracked files present (use "git add" to track)
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say "git add .?"
hint: Turn this message off by running
hint: git config advice.addemptypathspec false"
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git add
warning: adding embedded git repository: Cpp_Programs
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:   git submodule add <url> Cpp_Programs
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:   git rm --cached Cpp_Programs
hint:
hint: See "git help submodule" for more information.
warning: LF will be replaced by CRLF in test.cpp.
The file will have its original line endings in your working directory
bhawa@DESKTOP-H8M9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git status
On branch master
No commits yet

```



```
MINGW64/c/Users/bhawa/OneDrive/Desktop/clone
bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git commit -m "learning clone"
[master (root-commit) 5637cae] learning clone
2 files changed, 2 insertions(+)
create mode 160000 Cpp_Programs
create mode 100644 test.cpp

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git log
commit 5637cae3724a73d55a407c992fac2e3e8ca5f8 (HEAD -> master)
Author: Bhawana Yadav <bhawanayadav0367_ba21@chitkara.edu.in>
Date:   Sun Apr 10 22:18:19 2022 +0530

    learning clone

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git push -u origin master
fatal: 'origin' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git push master
fatal: 'master' does not appear to be a git repository
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git push
fatal: No configured push destination.
Either specify the URL from the command-line or configure a remote repository using
    git remote add <name> <url>
and then push using the remote name
    git push <name>

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git remote add origin "https://github.com/Bhawanayadav7/Cpp_Programs.git"
bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/clone (master)
$ git push -u origin master
Enumerating objects: 3, done.
```

Now we can see that the repository is cloned successfully after fork.

Task 1.2

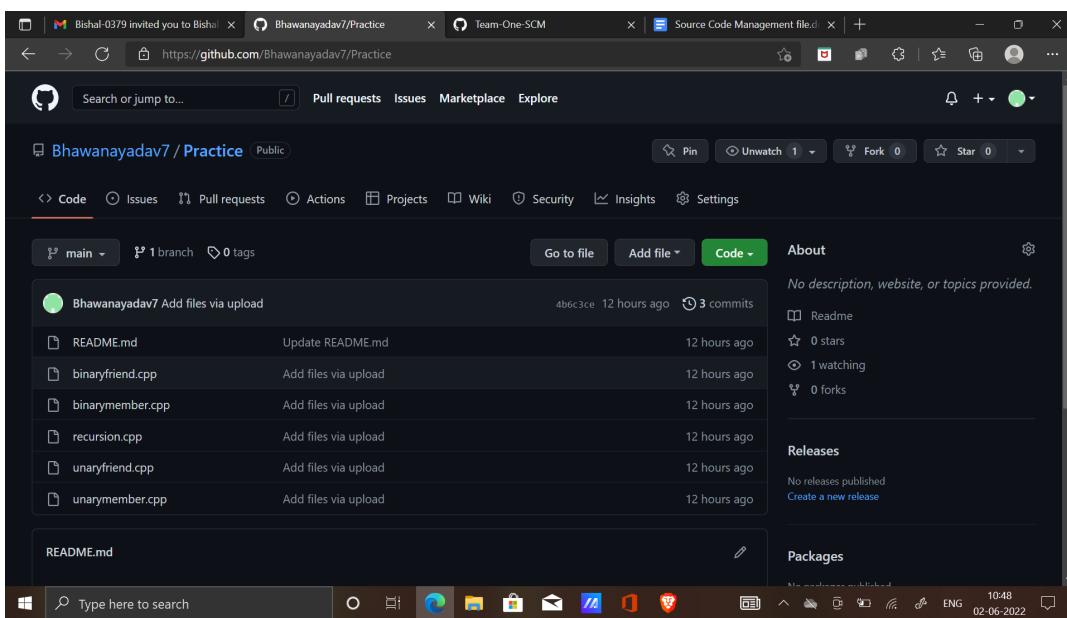
Aim : Add collaborators on GitHub.

Why add collaborators?

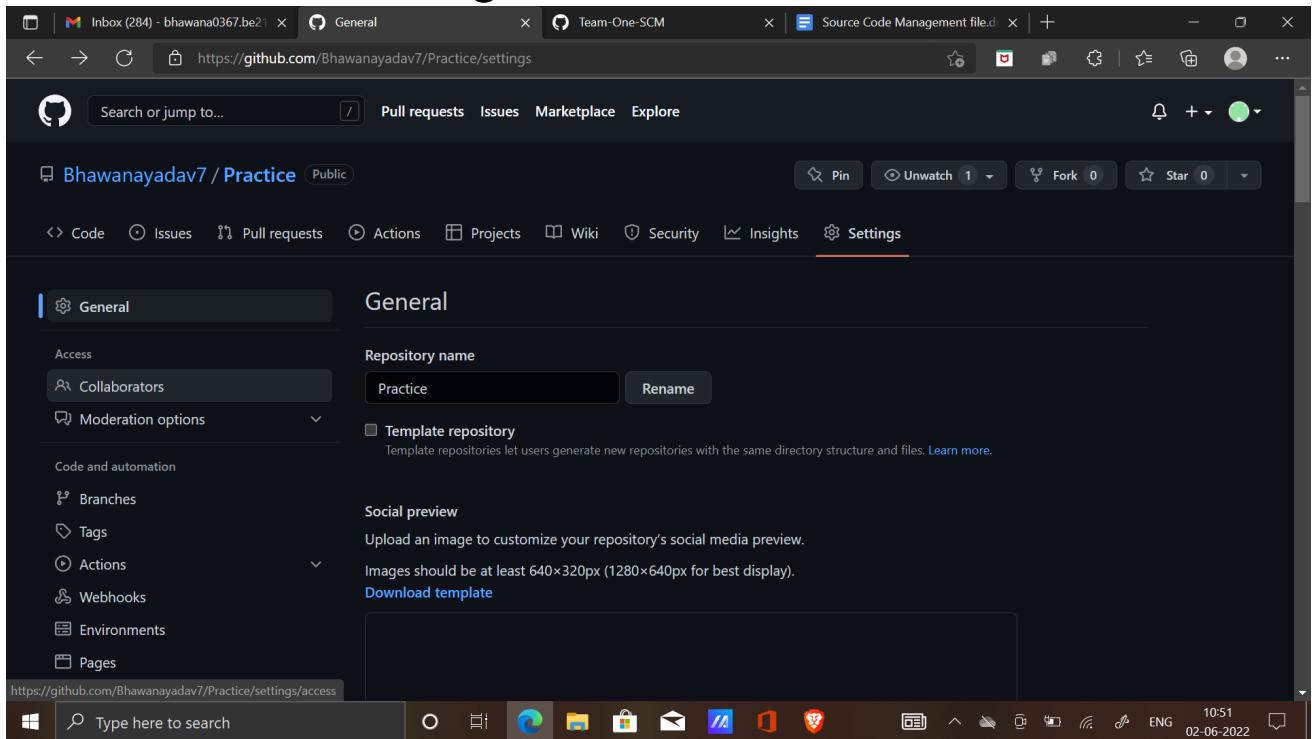
In GitHub, we can invite other GitHub users to become collaborators to our private repositories (which expires after 7 days if not accepted, restoring any unclaimed licenses). Being a collaborator of a personal repository you can pull (read) the contents of the repository and push (write) changes to the repository.

The following are the steps to invite other members to collaborate with your repository.

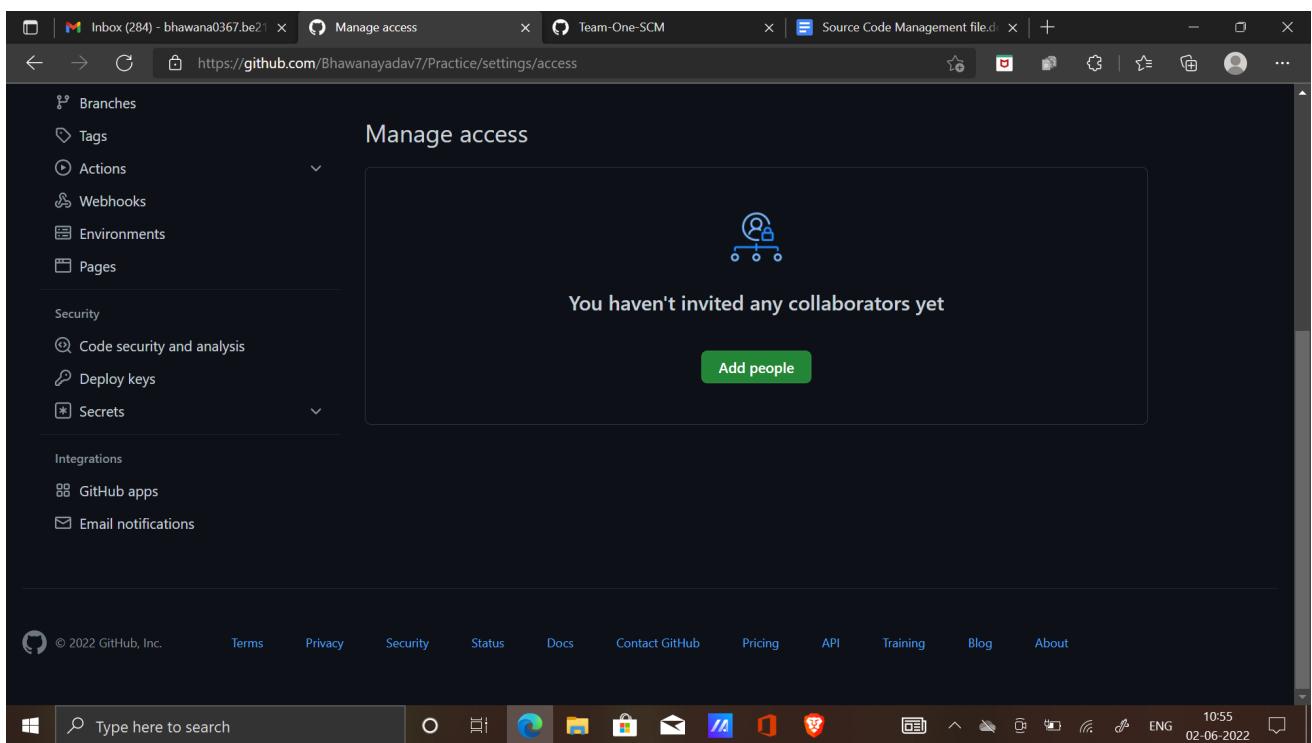
Step 1 : Click on the repository on which you want to add the collaborators then go to the Settings tab in the right corner of the GitHub page.



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

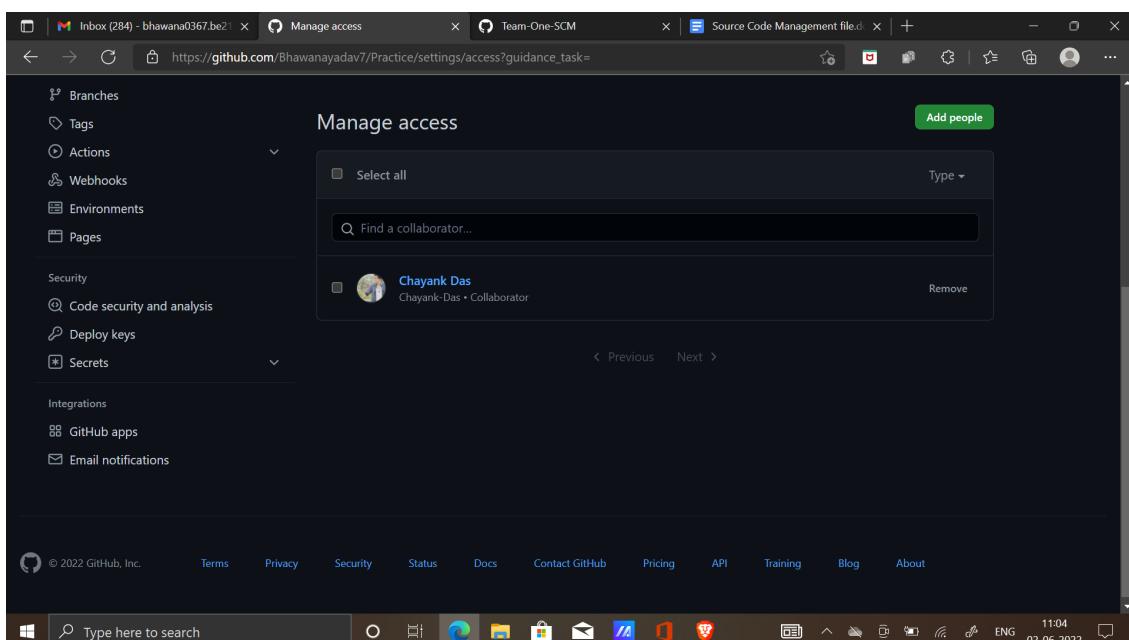


Step 3 : You can Invite collaborators by any of the following options —> Username, Full name or Email. After you send the invite, the collaborator receives an email invitation. The collaborator has to accept it in order to get permission to collaborate on the same project.



The screenshot shows the 'Manage access' page on GitHub. On the left, there's a sidebar with options like Branches, Tags, Actions, Webhooks, Environments, Pages, Security, Code security and analysis, Deploy keys, Secrets, Integrations, GitHub apps, and Email notifications. The main area is titled 'Manage access' and features a central icon of a person with a lock. Below it, the text 'You haven't invited any collaborators yet' is displayed, followed by a green 'Add people' button.

Under the **Manage access** we can view who has or hasn't accepted the collaborator invitation. Here in the below screen you can see chayank has accepted the request and is successfully a collaborator.



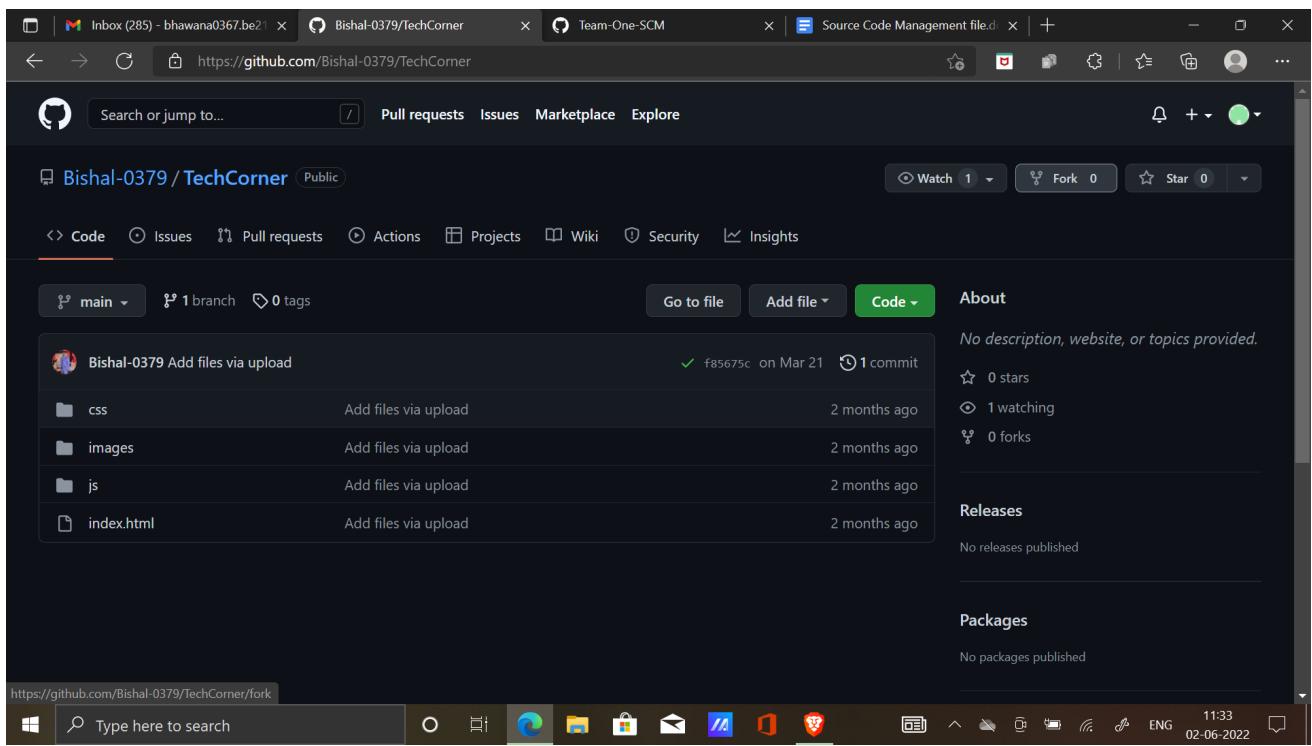
The screenshot shows the 'Manage access' page on GitHub, similar to the previous one but with a different result. The sidebar and main interface are identical, but the main area now lists 'Chayank Das' as a collaborator. The 'Select all' checkbox is checked, and there's a 'Remove' link next to Chayank Das's name. Navigation arrows at the bottom indicate 'Previous' and 'Next' pages.

Aim: Fork And Commit

Fork is a copy of an original repository in which you can make changes without affecting the original repository.

The following steps should be performed to fork and commit changes in your repository:-

Step 1 : To fork a repository, Click on the Fork button as shown below



Step 2 : After forking the repository, You need to add the forked repo to your account by clicking on it.



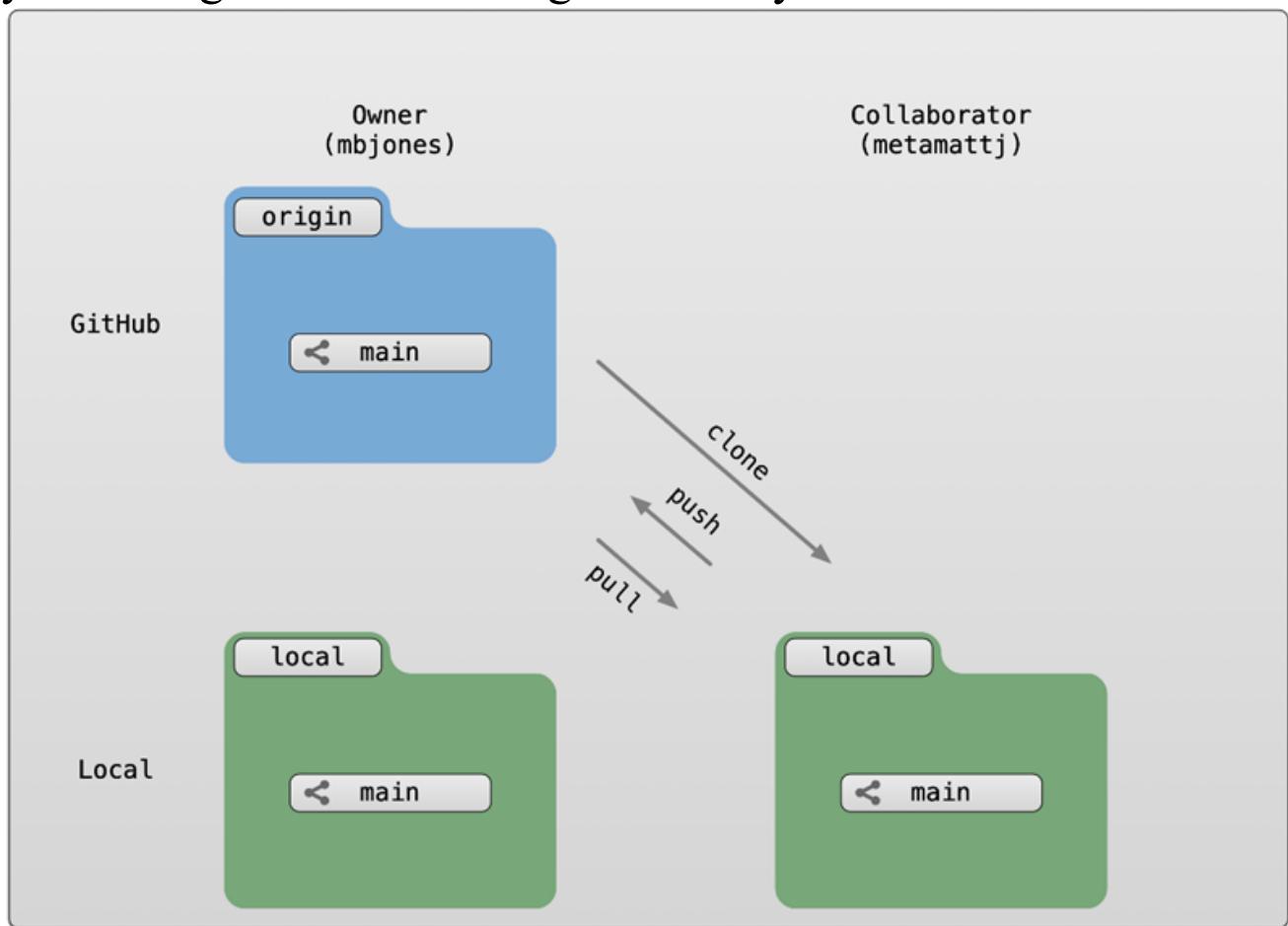
A screenshot of a Microsoft Edge browser window displaying a GitHub repository page. The repository is named "Bhawanayadav7/TechCorner" and is public. It was forked from "Bishal-0379/TechCorner". The "Code" tab is selected, showing the "main" branch with one commit by "Bishal-0379" titled "Add files via upload". The commit details show three files added: "css", "images", and "js", all updated "2 months ago". A message at the bottom encourages adding a README. The GitHub interface includes standard navigation links like Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. On the right side, there are sections for About (forked from Bishal Das), Releases (no releases published), and Packages (no packages published). The bottom of the screen shows the Windows taskbar with various pinned icons.

Step 3 : After the forking process is complete, you can commit changes in the file.

A screenshot of a Microsoft Edge browser window showing the same GitHub repository page. This time, the "Code" tab is selected, and the user is editing the "README.md" file. The file content is "# TechCorner" and "Forked From Bishal Das.". The GitHub interface remains consistent with the previous screenshot, including the sidebar with "About", "Releases", and "Packages" sections. The bottom of the screen shows the Windows taskbar.

Aim : Merge and Resolve conflicts created due to own activity and collaborators activity.

Git is a great tool for working on your own, but even better for working with friends and colleagues. Git allows you to work with confidence on your own local copy of files with the confidence that you will be able to successfully synchronize your changes with the changes made by others.

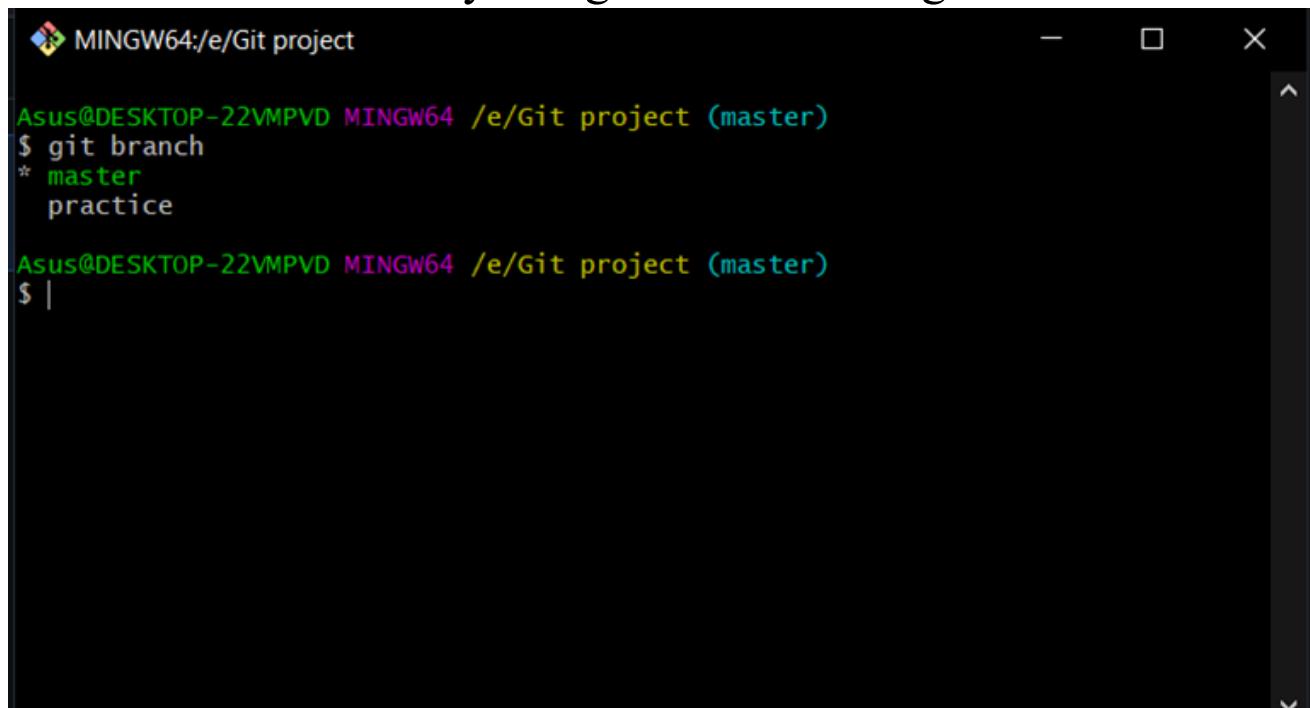


Git Merge

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 a 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.

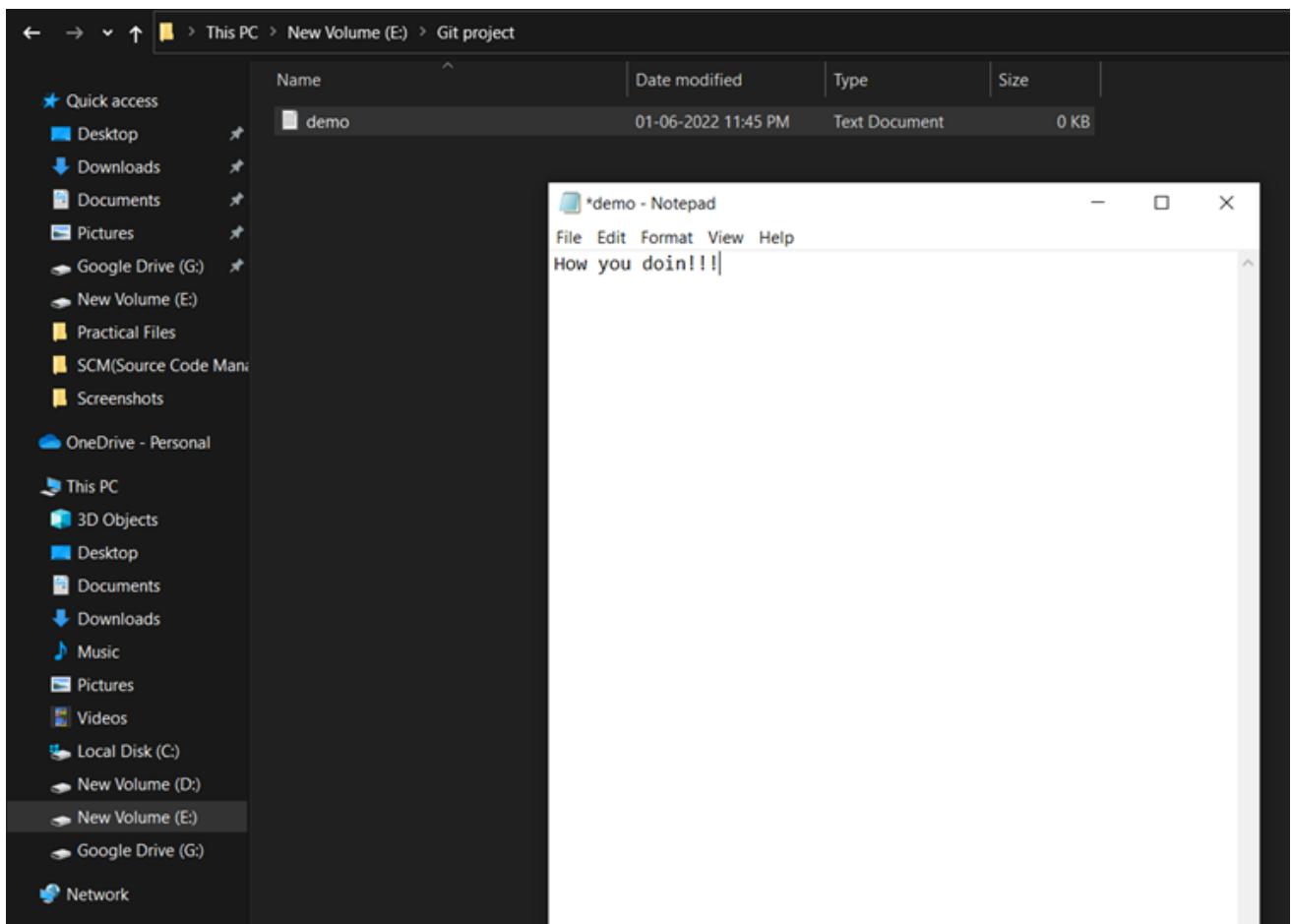


A screenshot of a terminal window titled "MINGW64:/e/Git project". The window shows the command \$ git branch being run, with the output displaying two branches: "master" (marked with an asterisk) and "practice".

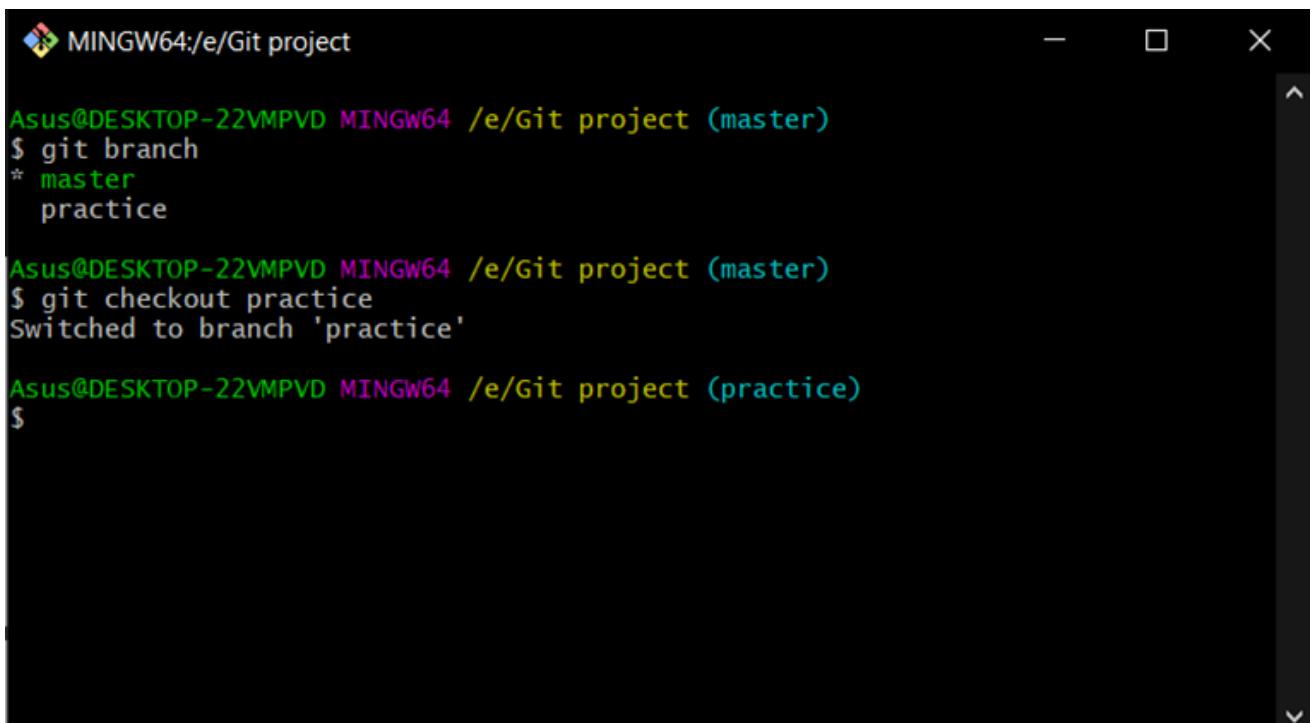
```
Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (master)
$ git branch
* master
  practice

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (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.



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

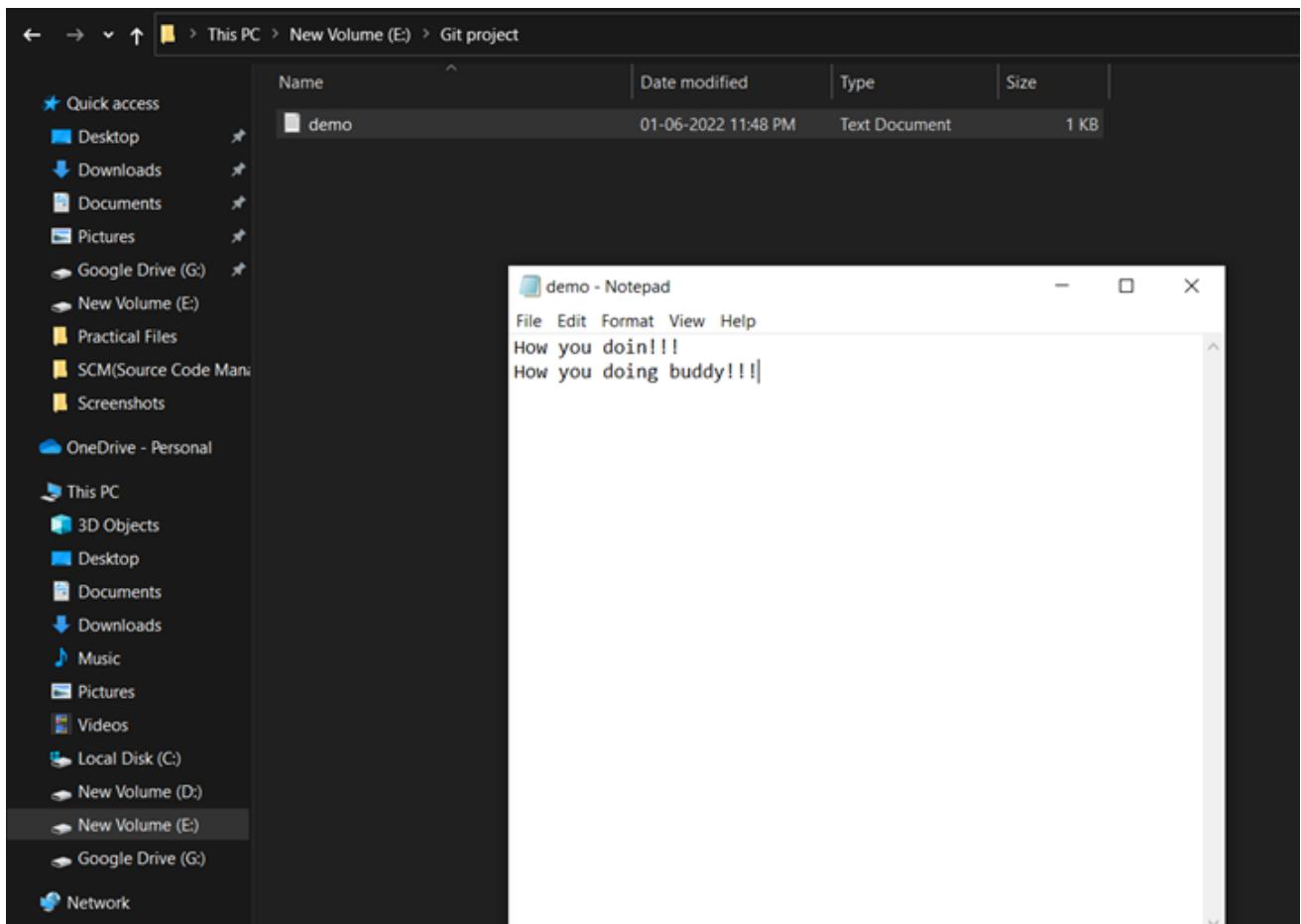


MINGW64:/e/Git project

```
Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (master)
$ git branch
* master
  practice

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (master)
$ git checkout practice
Switched to branch 'practice'

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (practice)
$
```



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

```
MINGW64:/e/Git project
Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (master)
$ git branch
* master
  practice

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (master)
$ git checkout practice
Switched to branch 'practice'

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (practice)
$ git add --
warning: CRLF will be replaced by LF in demo.txt.
The file will have its original line endings in your working directory
Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (practice)
$ git add --

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (practice)
$ git commit -m "Second commit"
[practice 3fc76f4] Second commit
 1 file changed, 2 insertions(+)
Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (practice)
$ git push --all
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 275 bytes | 275.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'practice' on GitHub by visiting:
remote:     https://github.com/Bishal-0379/git-project/pull/new/practice
remote:
To https://github.com/Bishal-0379/git-project.git
 * [new branch]      practice -> practice

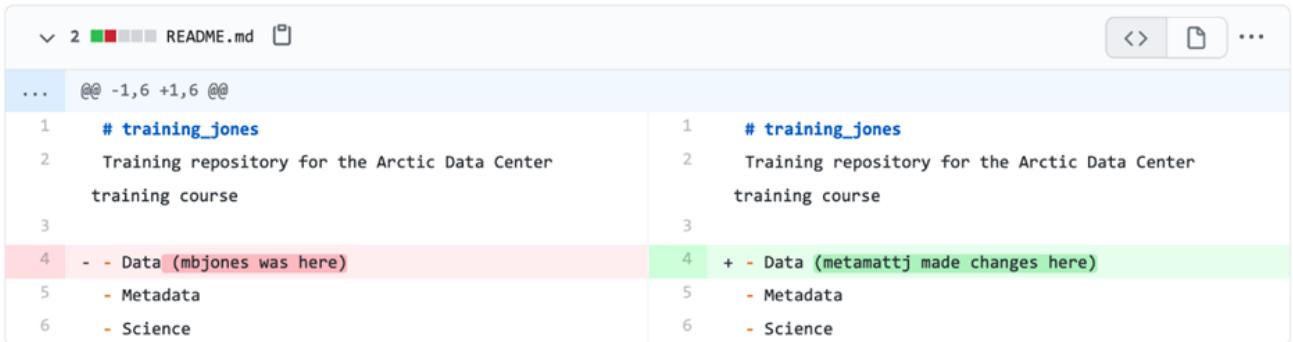
Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (practice)
$ git checkout master
Switched to branch 'master'

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (master)
$ git merge practice
Updating dec8157..3fc76f4
Fast-forward
  demo.txt | 2 ++
  1 file changed, 2 insertions(+)

Asus@DESKTOP-22VMPVD MINGW64 /e/Git project (master)
$
```

Merge conflicts

A merge conflict occurs when both the owner and collaborator change the same lines in the same file without first pulling the changes that the other has made. This is most easily avoided by good communication about who is working on various sections of each file, and trying to avoid overlaps. But sometimes it happens, and git is there to warn you about potential problems. And git will not allow you to overwrite one person's changes to a file with another's changes to the same file if they were based on the same version.



The screenshot shows a GitHub pull request diff for a file named README.md. The diff highlights changes made by two users: 'training_jones' and 'metamattj'. A conflict is visible at line 4, where both users have modified the same line. The original line was '- Data (mbjones was here)'. User 'training_jones' changed it to '+ Data (metamattj made changes here)', while user 'metamattj' changed it back to '- Data (mbjones was here)'. The code editor interface includes standard GitHub icons for file operations like copy and paste.

The main problem with merge conflicts is that, when the Owner and Collaborator both make changes to the same line of a file, git doesn't know whose changes take precedence. You have to tell git whose changes to use for that line.

How to resolve a conflict

- **Abort, abort, abort...**

Sometimes you just made a mistake. When you get a merge conflict, the repository is placed in a ‘Merging’ state until you resolve it. There’s a commandline command to abort doing the merge altogether:

git merge --abort

Of course, after doing that you still haven’t synced with your collaborator’s changes, so things are still unresolved. But at least your repository is now usable on your local machine.

· **Checkout**

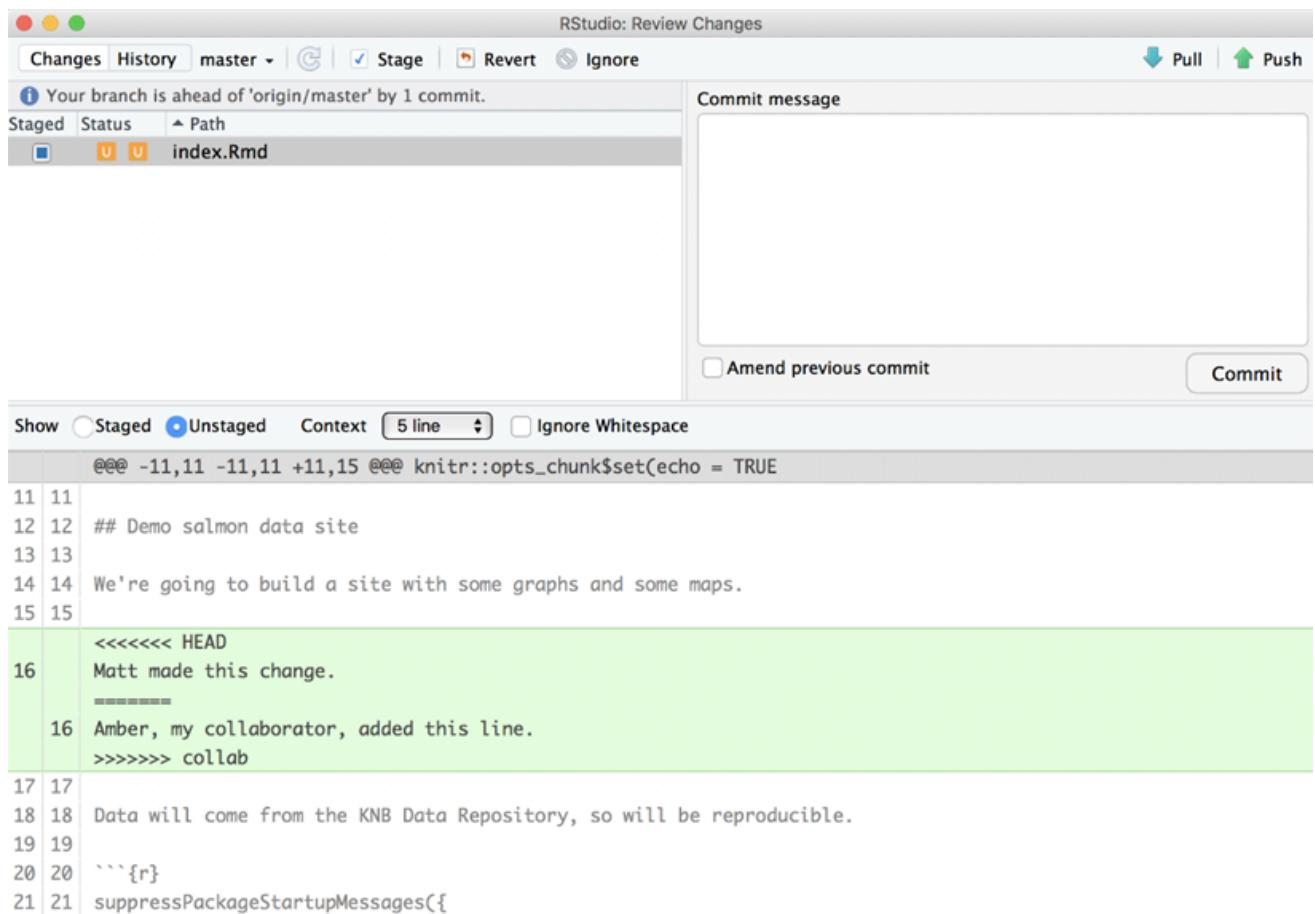
The simplest way to resolve a conflict, given that you know whose version of the file you want to keep, is to use the command line git program to tell git to use either your changes (the person doing the merge), or their changes (the other collaborator).

- keep your collaborators file: **git checkout --theirs conflicted_file.Rmd**
- keep your own file: **git checkout --ours conflicted_file.Rmd**

Once you have run that command, then run add, commit, and push the changes as normal.

- **Pull and edit the file**

But that requires the command line. If you want to resolve from RStudio, or if you want to pick and choose some of your changes and some of your collaborator's, then instead you can manually edit and fix the file. When you pulled the file with a conflict, git notices that there is a conflict and modifies the file to show both your own changes and your collaborator's changes in the file. It also shows the file in the Git tab with an orange U icon, which indicates that the file is Unmerged, and therefore awaiting your help to resolve the conflict. It delimits these blocks with a series of less than and greater than signs, so they are easy to find:



The screenshot shows the RStudio interface for reviewing changes. The title bar says "RStudio: Review Changes". The top menu has "Changes", "History", "master", "Stage", "Revert", and "Ignore". On the right, there are "Pull" and "Push" buttons. A message in the top left says "Your branch is ahead of 'origin/master' by 1 commit." Below it, a table shows "Staged" and "Status" for "index.Rmd", with "U" indicating untracked status. To the right is a "Commit message" field with a "Commit" button below it. A checkbox for "Amend previous commit" is also present. At the bottom, there are "Show" buttons for "Staged" (selected) and "Unstaged", a "Context" dropdown set to "5 line", and an "Ignore Whitespace" checkbox.

```

@@@ -11,11 -11,11 +11,15 @@@ knitr::opts_chunk$set(echo = TRUE
11 11
12 12 ## Demo salmon data site
13 13
14 14 We're going to build a site with some graphs and some maps.
15 15
16 <<<<< HEAD
16 Matt made this change.
16 =====
16 Amber, my collaborator, added this line.
16 >>>>> collab
17 17
18 18 Data will come from the KNB Data Repository, so will be reproducible.
19 19
20 20 `~`{r}
21 21 suppressPackageStartupMessages({

```

To resolve the conflicts, simply find all of these blocks, and edit them so that the file looks how you want (either pick your lines, your collaborators lines, some combination, or something altogether new), and save. Be sure you removed the delimiter lines that started with <<<<<<, =====, and >>>>>>.

Once you have made those changes, you simply add, commit, and push the files to resolve the conflict.

Workflows to avoid merge conflicts

Some basic rules of thumb can avoid the vast majority of merge conflicts, saving a lot of time and frustration. These are words our teams live by:

- Communicate often
- Tell each other what you are working on
- Pull immediately before you commit or push
- Commit often in small chunks.

A good workflow is encapsulated as follows:

Pull -> Edit -> Add -> Pull -> Commit -> Push

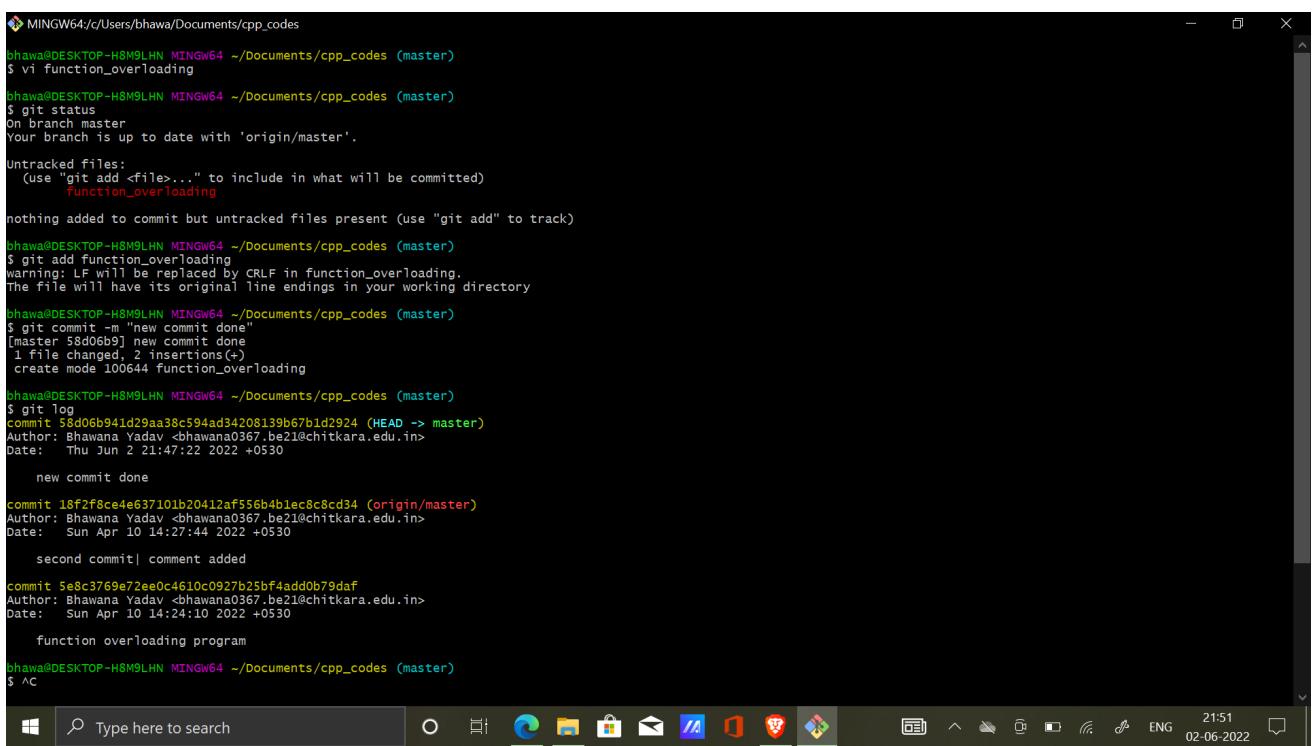
Always start your working sessions with a pull to get any outstanding changes, then start doing your editing and work. Stage your changes, but before you commit, Pull again to see if any new changes have arrived. If so, they should merge in easily if you are working in different parts of the program. You can then Commit and immediately Push your changes safely. Good luck, and try to not get frustrated. Once you figure out how to handle merge conflicts, they can be avoided or dispatched when they occur, but it does take a bit of practice.

Aim: Reset and Revert

What is git revert?

The git revert command is an “undo” operation however it is not the appropriate one. The git revert command reverts the changes introduced by the commit and appends a new commit with resulting reversed content. This does not allow Git to lose history which is essential for revision history integrity and proper collaboration. Reverting is used for applying the inverse commit from the project history. You can use git revert for automatically going back and making fixes.

we made a new file function_overloading and committed changes then ran the command git revert with the git log commit id . see the screenshot attached below.



```

MINGW64:/c/Users/bhawa/Documents/cpp_codes
$ vim function_overloading
bhawa@DESKTOP-HSM9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

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

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

bhawa@DESKTOP-HSM9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git add function_overloading
warning: LF will be replaced by CRLF in function_overloading.
The file will have its original line endings in your working directory

bhawa@DESKTOP-HSM9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git commit -m "new commit done"
[master 58d06b9] new commit done
 1 file changed, 2 insertions(+)
 create mode 100644 function_overloading

bhawa@DESKTOP-HSM9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git log
commit 58d06b941d29aa38c594ad34208139b67b1d2924 (HEAD -> master)
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Thu Jun 2 21:47:22 2022 +0530

  new commit done

commit 18f2f8ce4e637101b20412af556b4b1ec8c8cd34 (origin/master)
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Sun Apr 10 14:27:44 2022 +0530

  second commit| comment added

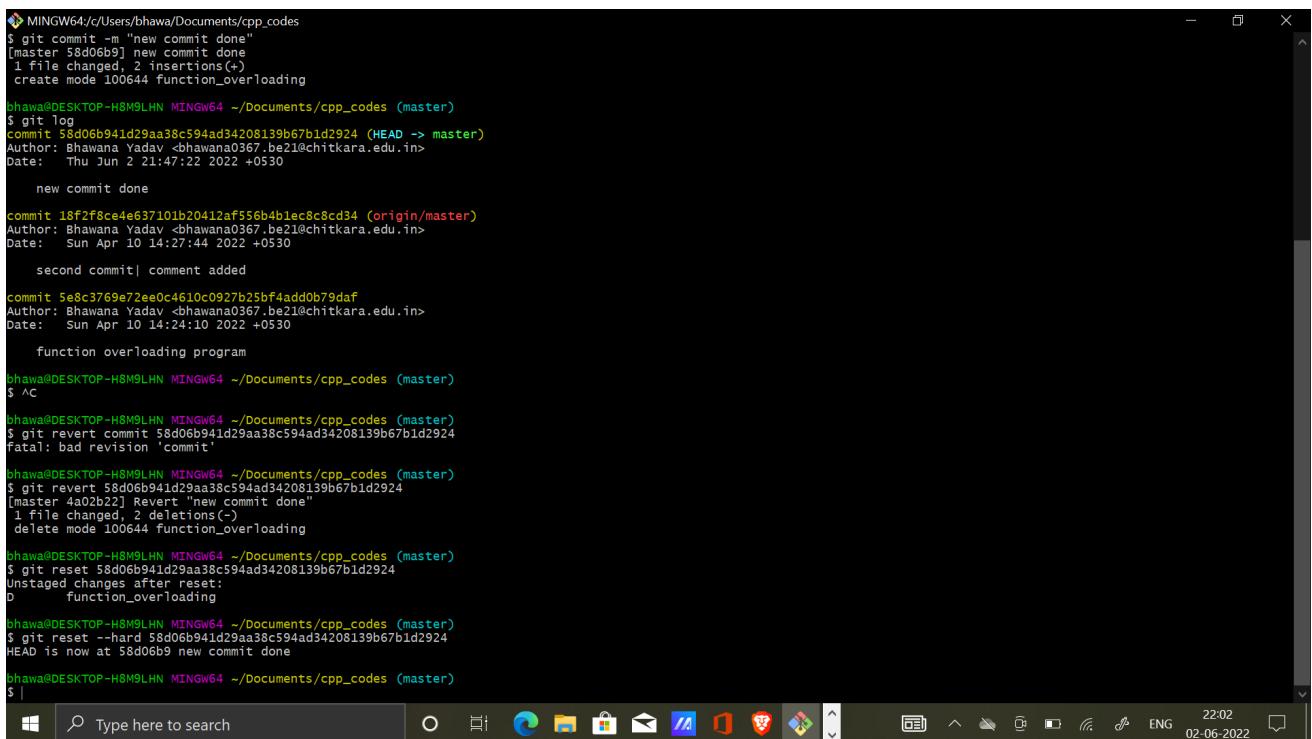
commit 5e8c3769e72ee0c4610c0927b25bf4add0b79daf
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Sun Apr 10 14:24:10 2022 +0530

  function overloading program

bhawa@DESKTOP-HSM9LHN MINGW64 ~/Documents/cpp_codes (master)
$ ^C

```


The term reset stands for undoing changes. The git reset command is used to reset the changes.



```
MINGW64:/Users/bhawa/Documents/cpp_codes
$ git commit -m "new commit done"
[master 58d06b9] new commit done
 1 file changed, 2 insertions(+)
 create mode 100644 function_overloading

bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git log
commit 58d06b941d29aa38c594ad34208139b67b1d2924 (HEAD -> master)
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Thu Jun 2 21:47:22 2022 +0530

    new commit done

commit 18f2f8ce4e637101b20412af556b4b1ec8c8cd34 (origin/master)
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Sun Apr 10 14:27:44 2022 +0530

    second commit| comment added

commit 5e8c3769e72ee0c4610c0927b25bf4add0b79daf
Author: Bhawana Yadav <bhawana0367.be21@chitkara.edu.in>
Date:   Sun Apr 10 14:24:10 2022 +0530

    Function overloading program

bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ AC

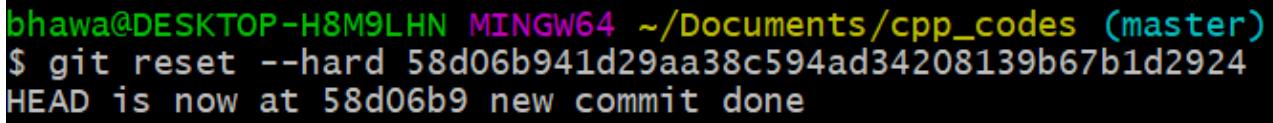
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git revert commit 58d06b941d29aa38c594ad34208139b67b1d2924
fatal: bad revision 'commit'

bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git revert 58d06b941d29aa38c594ad34208139b67b1d2924
[master 4a02b22] Revert "new commit done"
 1 file changed, 2 deletions(-)
 delete mode 100644 function_overloading

bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git reset --hard 58d06b941d29aa38c594ad34208139b67b1d2924
HEAD is now at 58d06b9 new commit done

bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ |
```

The last command is git reset where I have reseted the changes I had made earlier.



```
bhawa@DESKTOP-H8M9LHN MINGW64 ~/Documents/cpp_codes (master)
$ git reset --hard 58d06b941d29aa38c594ad34208139b67b1d2924
HEAD is now at 58d06b9 new commit done
```

Project Work

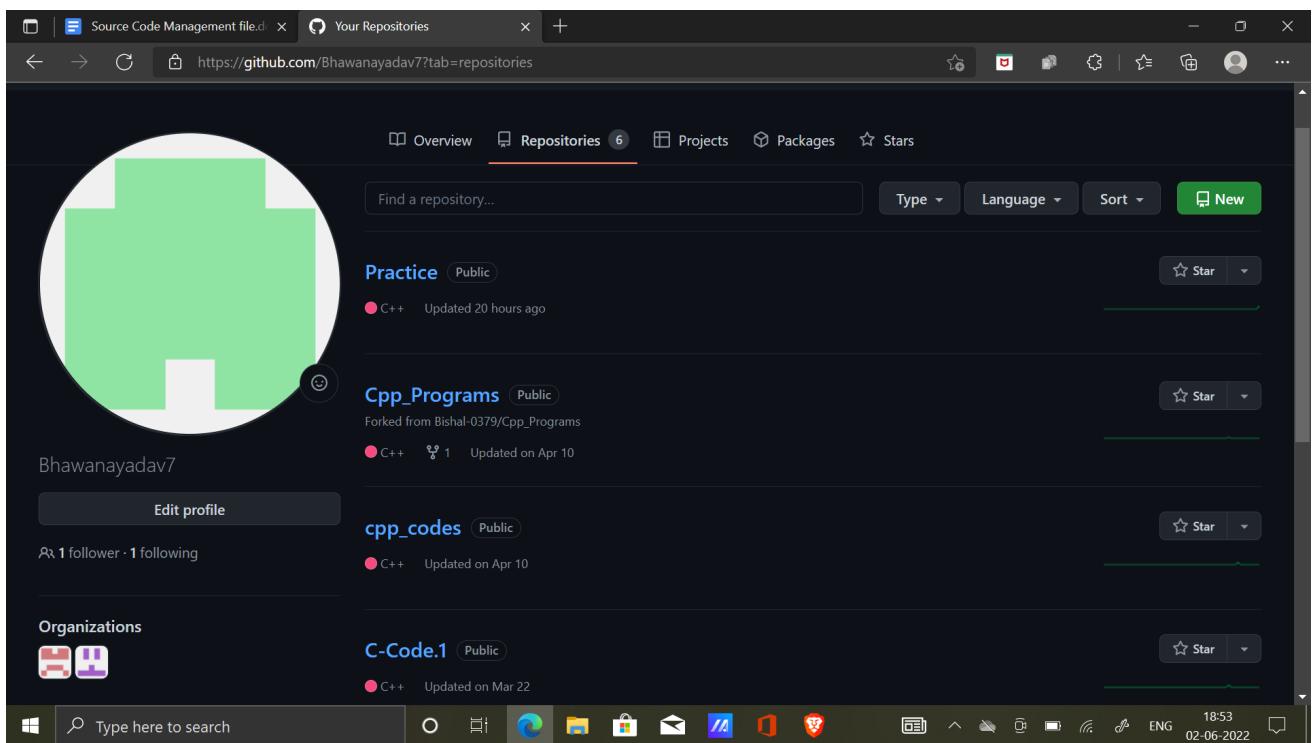
According to the given Task, each member has created a distributed repository and added team members, several tasks were done accordingly.

TEAM MEMBERS

Bhawana Yadav: 2110990367
Bishal Das: 2110990379
Chayank Das: 2110990388
Debarun Das: 2110990411

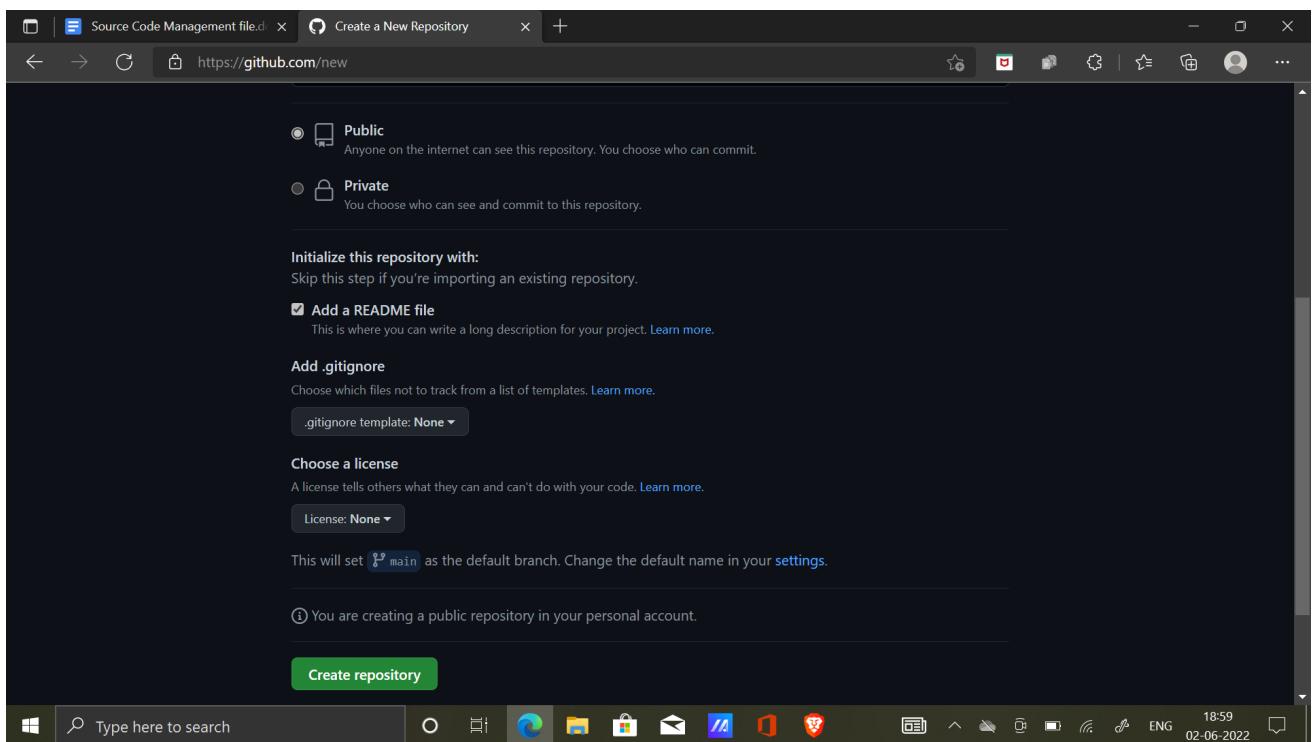
Aim: Create a distributed repository and add team members in the project team.

Step 1: Login to your Github account and you will land on the homepage as shown below. Click on Repositories option in the menu bar and go to new to create a new repository.



Step 2:

- Enter the Repository name and add description if you want.
- Select if you want the repository to be public or private.
- Click on “Create Repository”.



Step 3: click on settings and after that go to the collaborators option to add team members

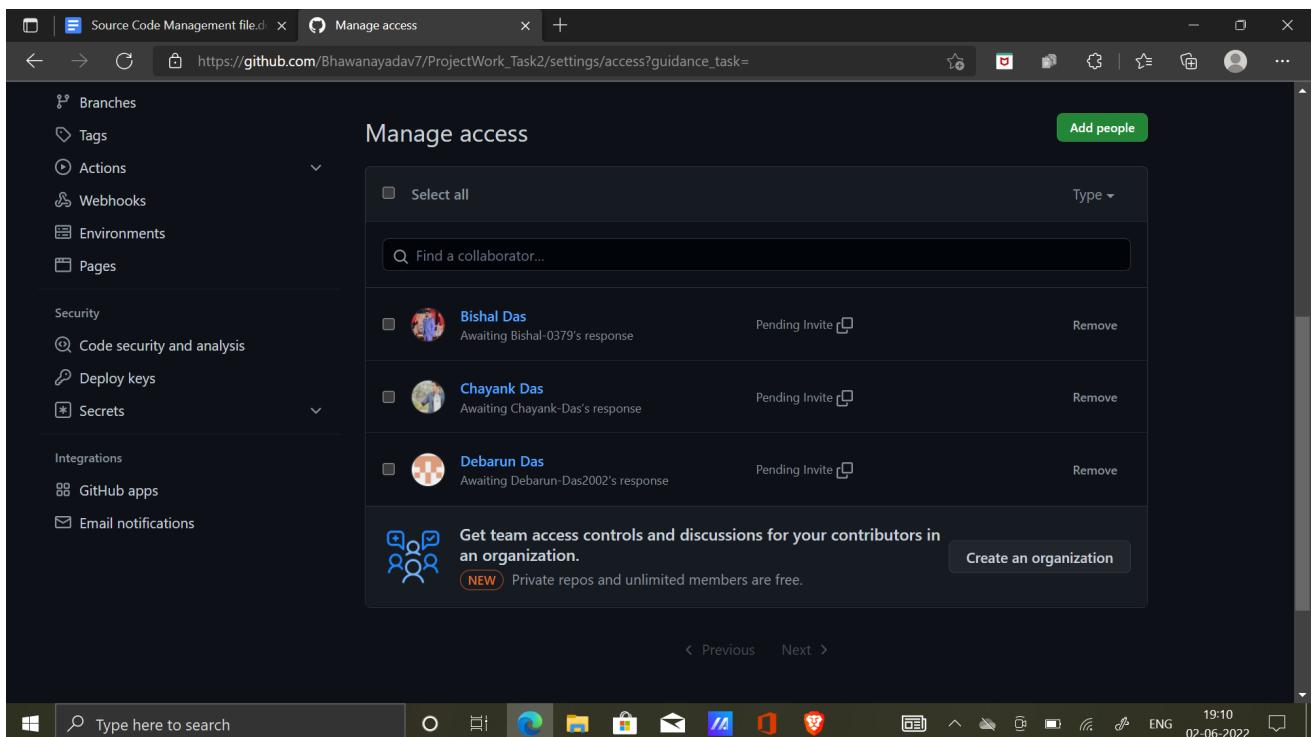


The screenshot shows a Microsoft Edge browser window with the URL https://github.com/Bhawanayadav7/ProjectWork_Task2/settings. The page is titled 'Bhawanayadav7 / ProjectWork_Task2' and is set to 'Public'. The 'General' tab is selected in the navigation bar. On the left, there's a sidebar with options like 'Access', 'Collaborators' (which is currently selected), 'Moderation options', 'Code and automation', 'Branches', 'Tags', 'Actions', 'Webhooks', 'Environments', and 'Pages'. The main content area shows the repository name 'ProjectWork_Task2' and a 'Rename' button. It also includes sections for 'Template repository' (with a note about generating new repositories) and 'Social preview' (with instructions to upload an image). At the bottom right of the main content area, there's a 'Download template' link. The status bar at the bottom of the browser shows the URL, the date '02-06-2022', and the time '19:02'.

Step 4: Add members by clicking on add people.

The screenshot shows the same Microsoft Edge browser window, now with the 'Manage access' tab selected in the navigation bar. The URL has changed to https://github.com/Bhawanayadav7/ProjectWork_Task2/settings/access. The sidebar remains the same. The main content area is titled 'Manage access' and features a message 'You haven't invited any collaborators yet' with a small user icon. Below this is a green 'Add people' button. The status bar at the bottom of the browser shows the URL, the date '02-06-2022', and the time '19:03'.

Step 5: under Manage access you can see who has or hasn't accepted your request.



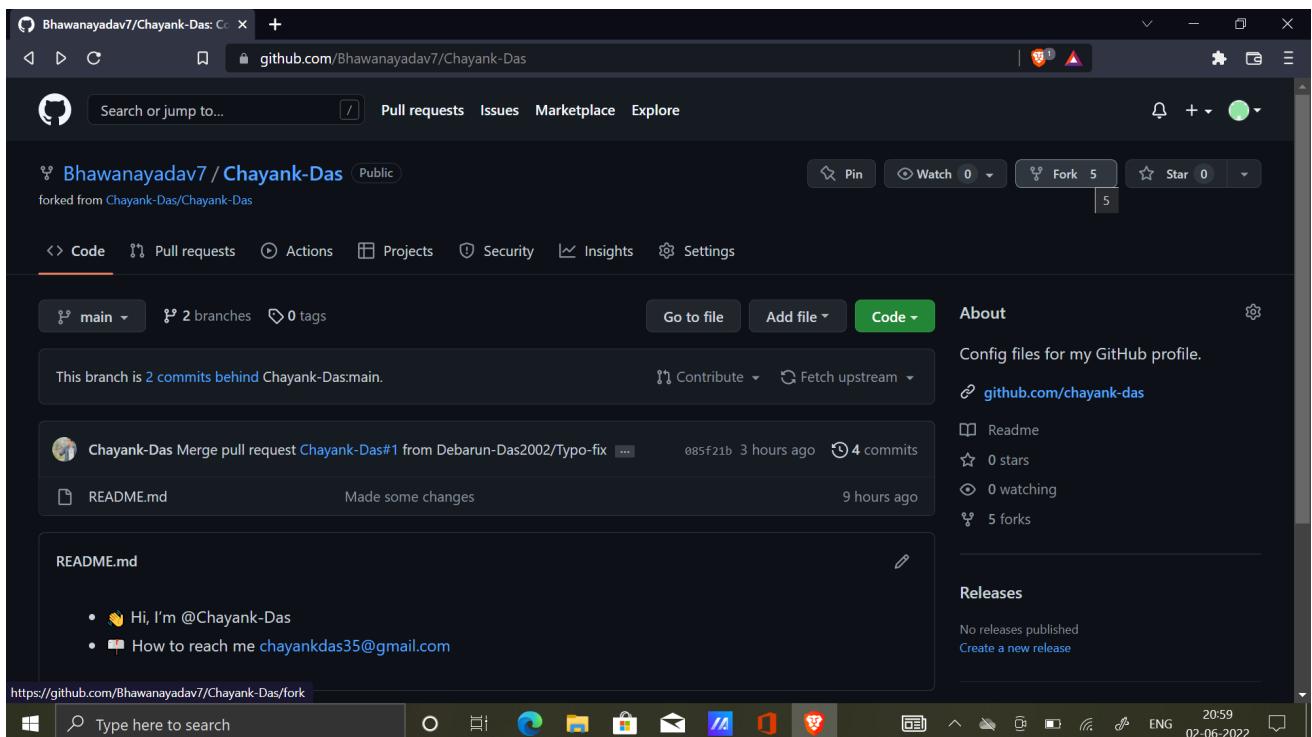
The screenshot shows the 'Manage access' section of a GitHub repository settings page. On the left, there's a sidebar with links for Branches, Tags, Actions, Webhooks, Environments, Pages, Security, Code security and analysis, Deploy keys, Secrets, Integrations, GitHub apps, and Email notifications. The main area is titled 'Manage access' and lists three users with pending invites:

User	Status	Action
Bishal Das	Awaiting Bishal-0379's response	Pending Invite <input type="button" value="Pending Invite"/>
Chayank Das	Awaiting Chayank-Das's response	Pending Invite <input type="button" value="Pending Invite"/>
Debarun Das	Awaiting Debarun-Das2002's response	Pending Invite <input type="button" value="Pending Invite"/>

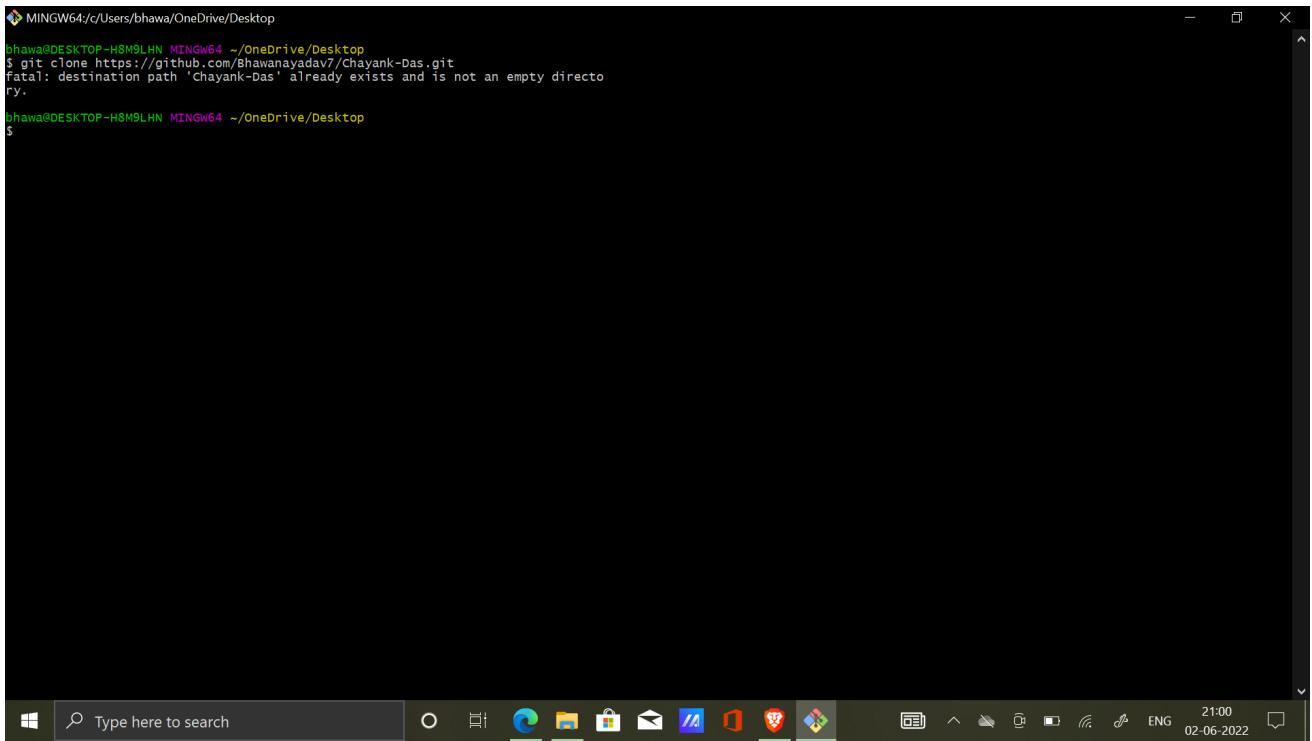
At the bottom of the main area, there's a call-to-action for creating an organization: 'Get team access controls and discussions for your contributors in an organization.' and 'Create an organization'. The taskbar at the bottom of the screen shows the Windows Start button, a search bar, and icons for File Explorer, Edge, Mail, and other pinned applications. The system tray shows the date and time as 02-06-2022 19:10.

Aim: Open and Close pull request on team members repository.

Step 1: Fork the repository you want to amend changes to.

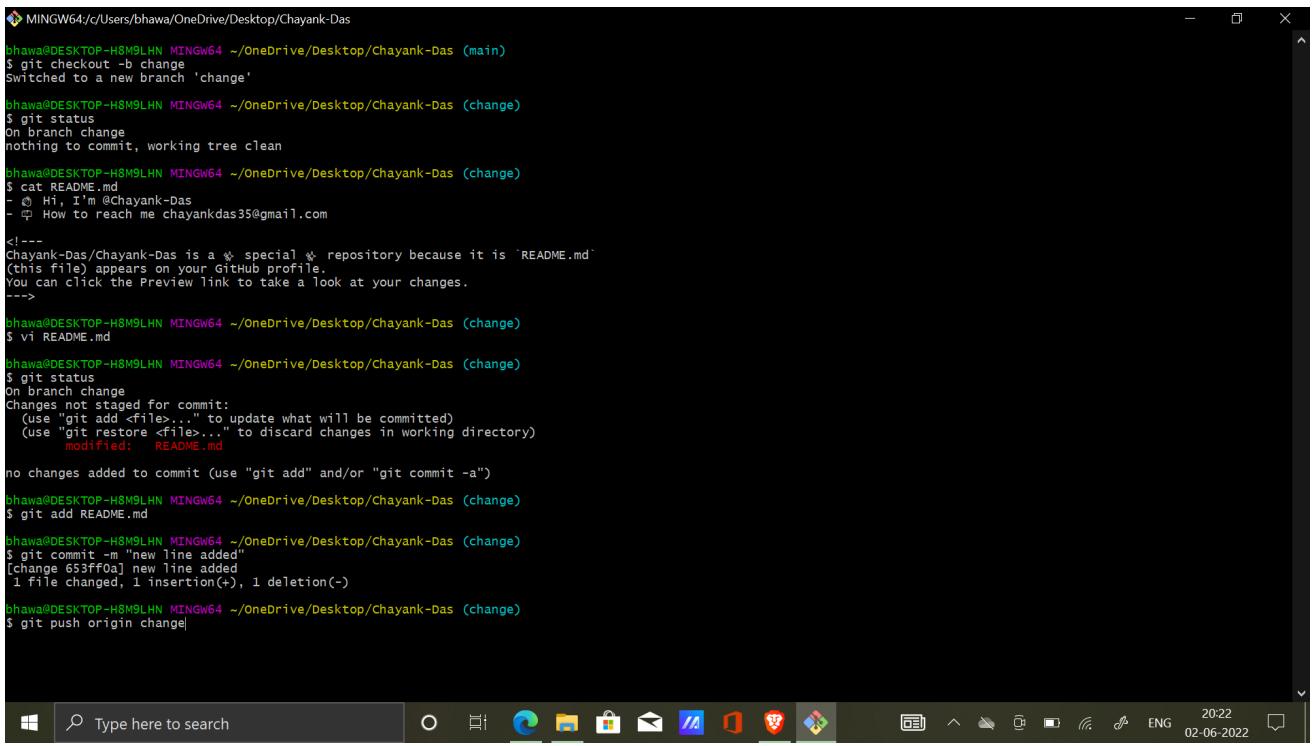


Step 2: Clone to your local computer.



```
MINGW64/c/Users/bhawa/OneDrive/Desktop
bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop
$ git clone https://github.com/Bhawanayadav7/Chayank-Das.git
fatal: destination path 'Chayank-Das' already exists and is not an empty directory.
bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop
$
```

Step 3: make some changes in any of the files or add new file and push it.



```
MINGW64/c/Users/bhawa/OneDrive/Desktop/Chayank-Das
bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (main)
$ git checkout -b change
Switched to a new branch 'change'

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (change)
$ git status
On branch change
nothing to commit, working tree clean

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (change)
$ cat README.md
Hi, I'm @chayank-das
How to reach me chayankdas35@gmail.com
<!---
Chayank-Das/Chayank-Das is a special repository because it is 'README.md'
(this file) appears on your GitHub profile.
You can click the Preview link to take a look at your changes.
-->

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (change)
$ vi README.md

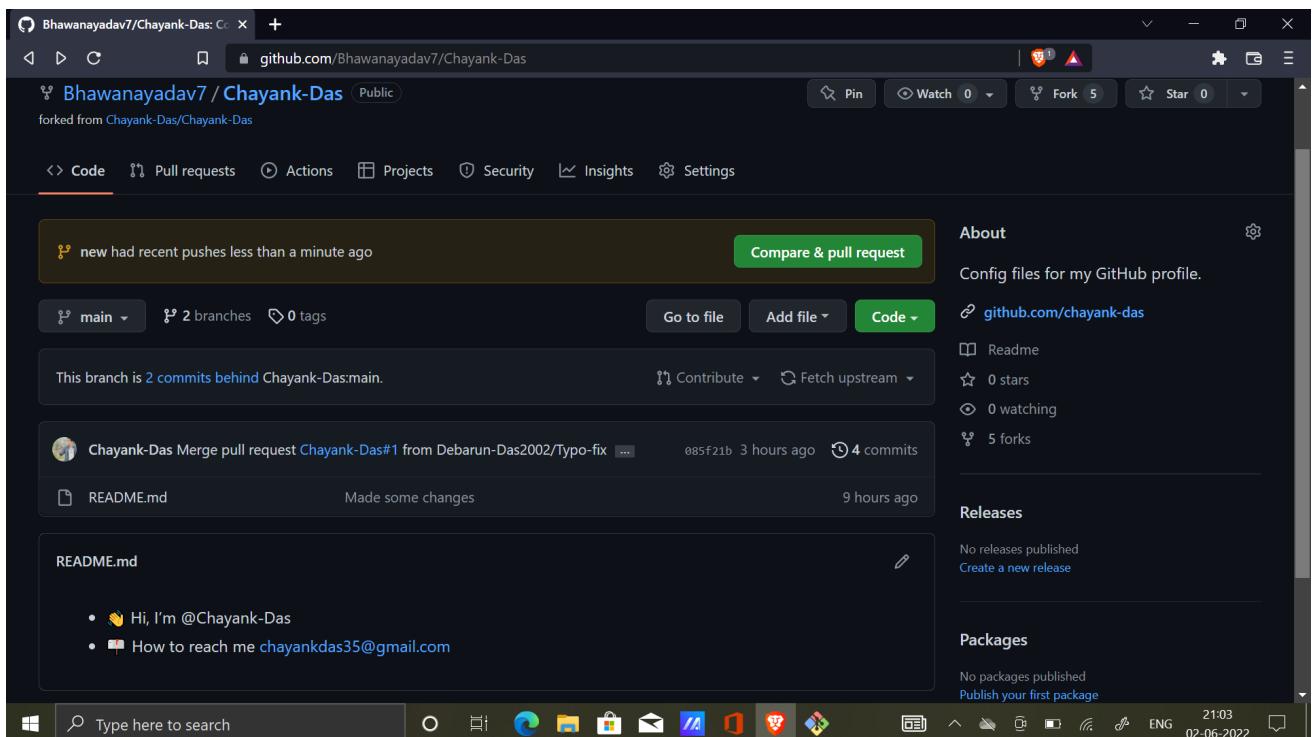
bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (change)
$ git status
On branch change
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified: README.md

no changes added to commit (use "git add" and/or "git commit -a")
bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (change)
$ git add README.md

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (change)
$ git commit -m "new line added"
[change 653ff0a] new line added
 1 file changed, 1 insertion(+), 1 deletion(-)

bhawa@DESKTOP-HSM9LHN MINGW64 ~/OneDrive/Desktop/Chayank-Das (change)
$ git push origin change
```

Step 4: compare and pull the merge request. As we are team members no open request will be needed.



Step 5: After the compare and pull request it is completely done.



Comparing Chayank-Das:main...B · +

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: Chayank-Das/Chayank-Das · base: main · head repository: Bhawanayadav7/Chayank-Das · compare: new

✓ Able to merge. These branches can be automatically merged.

\$ small change

Write Preview

Leave a comment

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

Allow edits by maintainers

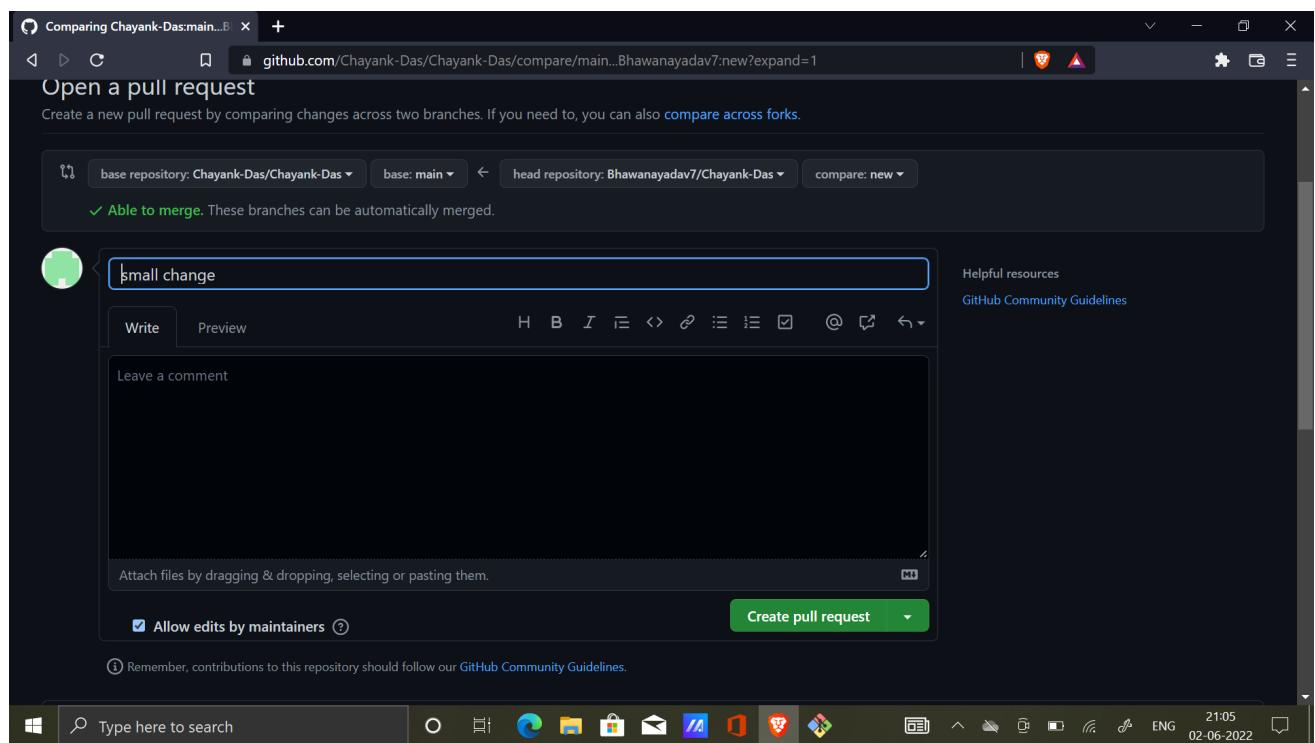
Create pull request

Helpful resources GitHub Community Guidelines

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

Type here to search

21:05 02-06-2022



small change by Bhawanayadav7 · +

Bhawanayadav7 wants to merge 1 commit into Chayank-Das:main from Bhawanayadav7:new

Conversation 0 · -> Commits 1 · Checks 0 · Files changed 1

Bhawanayadav7 commented now

No description provided.

small change

Add more commits by pushing to the new branch on Bhawanayadav7/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.

Reviewers

No reviews

Still in progress? Convert to draft

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

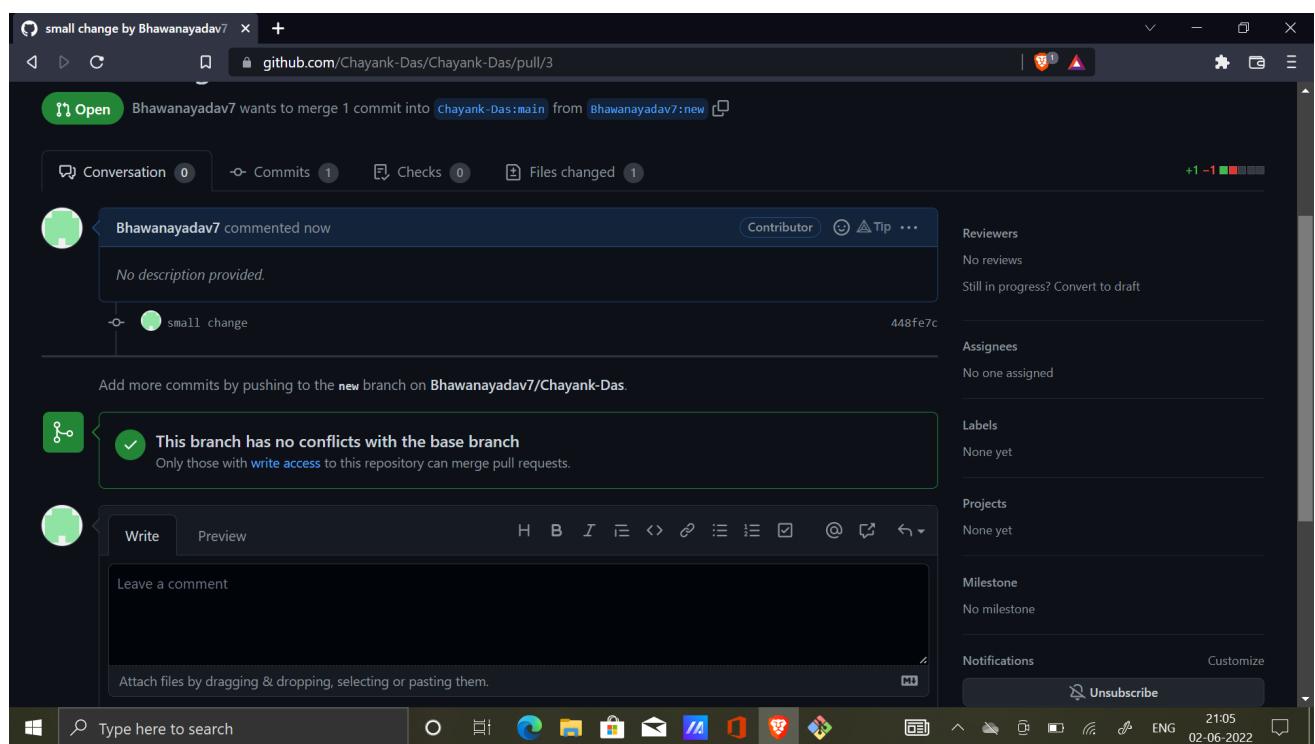
Notifications

Customize

Unsubscribe

Type here to search

21:05 02-06-2022



Aim: Publish and print Network graph

The network graph is one of the most useful feature for developers on GitHub. It is used to display the branch history of the entire repository network, including branches of the root repository and branches of forks that contain commits unique to the network.

A repository's graphs give you information on traffic, projects that depend on the repository, contributors and commits to the repository, and a repository's forks and network. If you maintain a repository, you can use this data to get a better understanding of who's using your repository and why they're using it.

Steps to access network graphs of respective repository

1. On [GitHub.com](#) navigate to the main page of the repository.
2. Under your repository name, click Insights.

Source Code Management file.d | Manage access | Bishal-0379 invited you to Bishal | Bhawanayadav7/ProjectWork_Task2

Search or jump to... Pull requests Issues Marketplace Explore

Bhawanayadav7 / ProjectWork_Task2 Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

Debarun-Das2002 Merge pull request #2 from Debarun-Das2002/fixedbyme ea47d10 27 minutes ago 6 commits

README.md Made some necessary changes 28 minutes ago

animal.exe Add files via upload 1 hour ago

README.md

ProjectWork_Task2

This is a distributed repository for task 2

This text is added by chavank this text is added by debarun

About

This is a distributed repository for task 2

Readme 0 stars 1 watching 2 forks

Releases

No releases published Create a new release

Packages

No packages published

https://github.com/Bhawanayadav7/ProjectWork_Task2/pulse

3. At the left sidebar, click on Network.

Source Code Management file.d | Manage access | Bishal-0379 invited you to Bishal | Pulse · Bhawanayadav7/ProjectWork_Task2

Search or jump to... Pull requests Issues Marketplace Explore

Bhawanayadav7 / ProjectWork_Task2 Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

May 26, 2022 – June 2, 2022 Period: 1 week

Overview

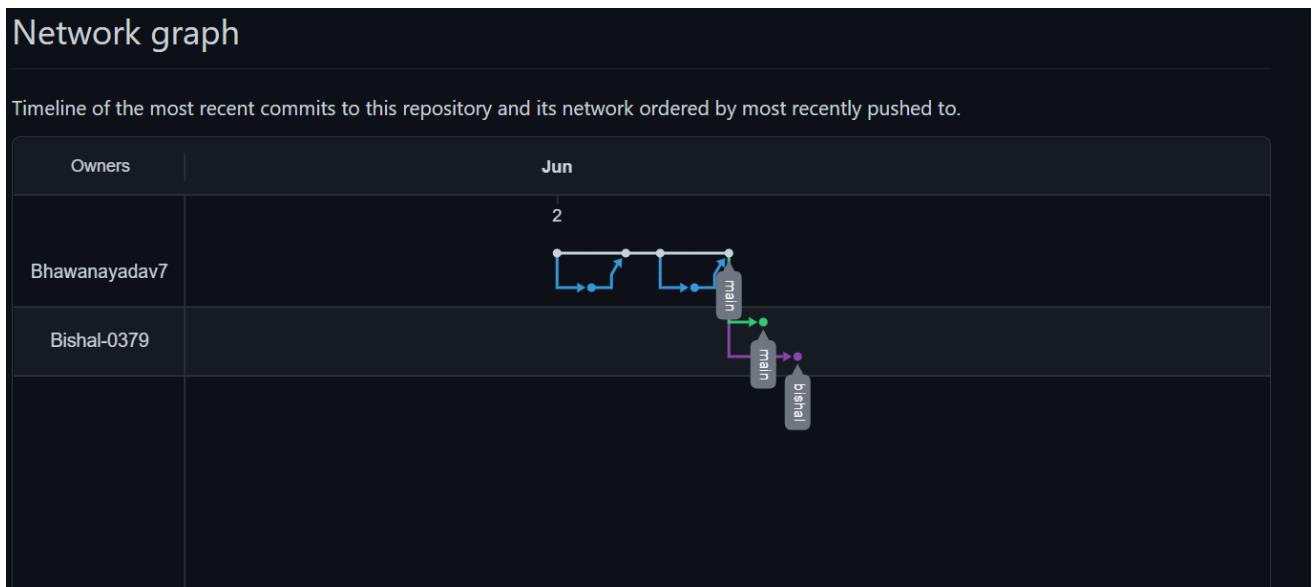
2 Active pull requests 0 Active issues

Merged pull requests: 2	Open pull requests: 0	Closed issues: 0	New issues: 0
-------------------------	-----------------------	------------------	---------------

Excluding merges, 3 authors have pushed 4 commits to main and 4 commits to all branches. On main, 0 files have changed and there have been 0 additions and 0 deletions.

https://github.com/Bhawanayadav7/ProjectWork_Task2/network

You will get the network graph of your repository which displays the branch history of the entire repository network, including branches of the root repository and branches of forks that contain commits unique to the network.



This is the network graph.