File

of

Source Code Management

by

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

CHITKARA UNIVERSITY

Index

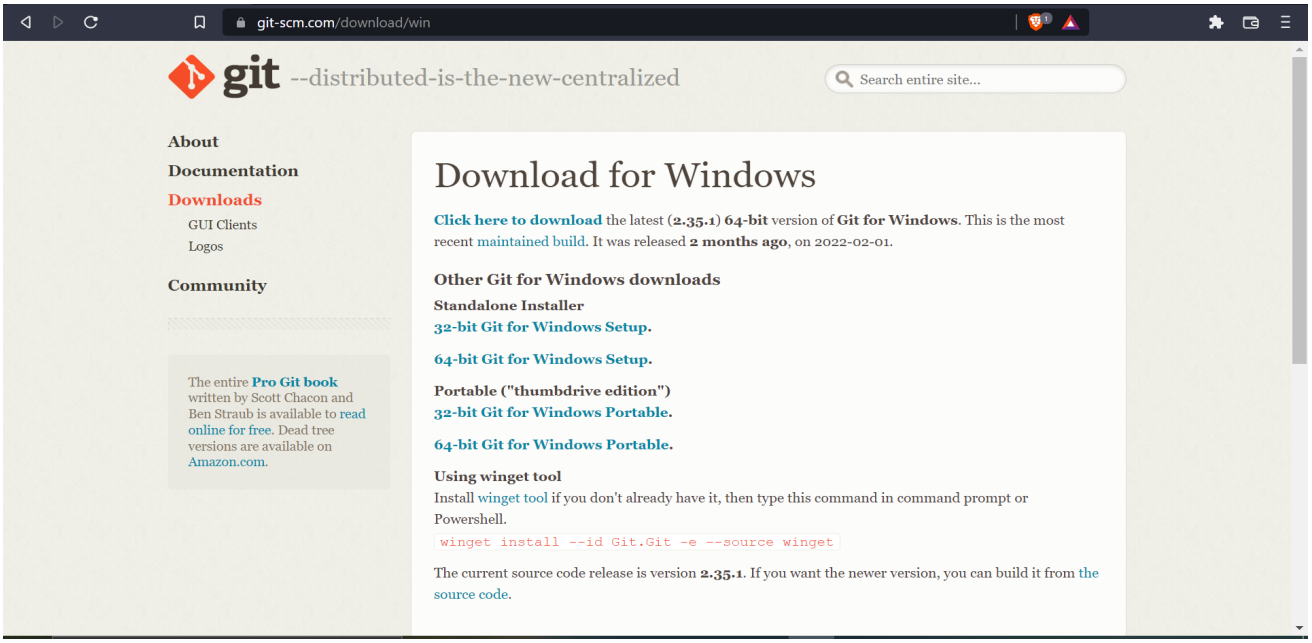
CHITKARA UNIVERSITY

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.

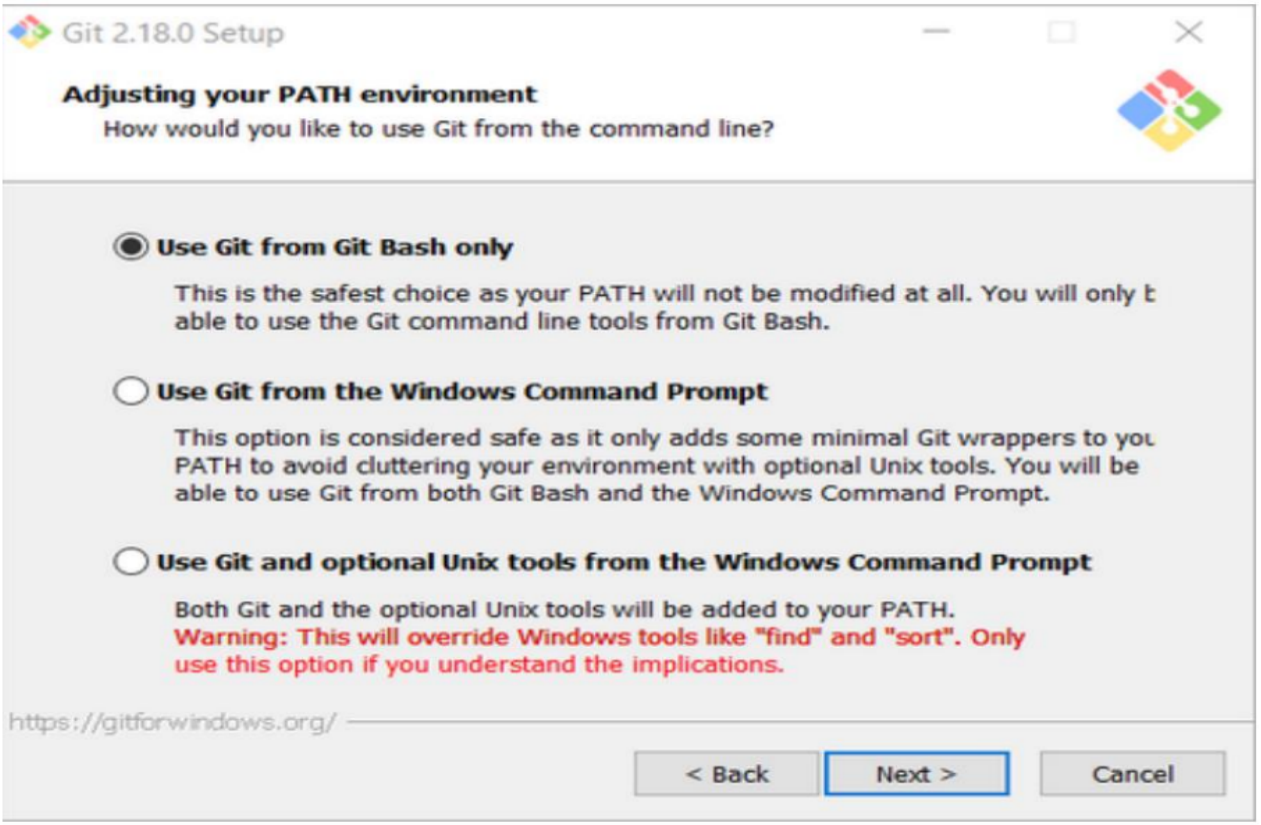


I will recommend going for default settings.
In the Adjusting your PATH screen, all three options are acceptable:

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.



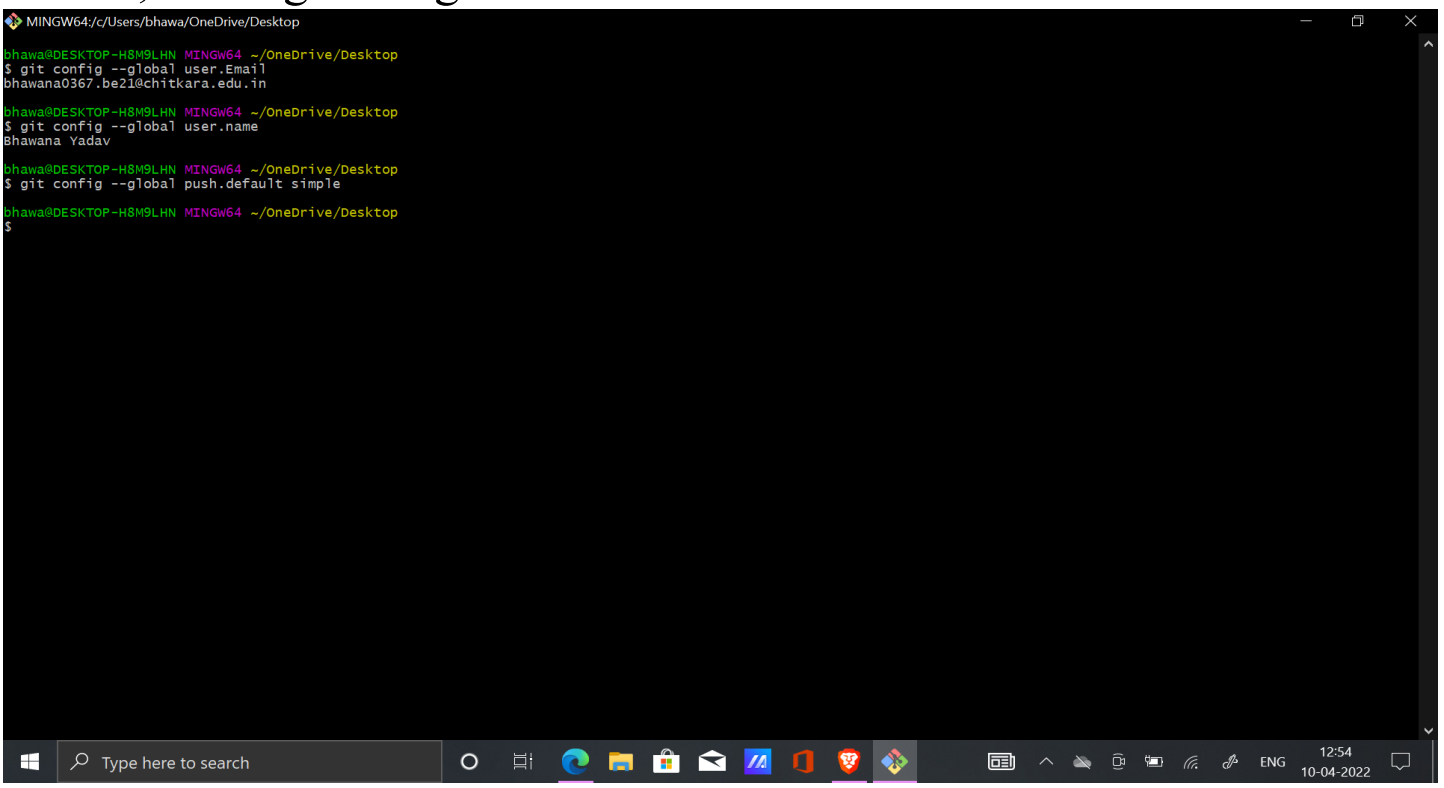
After the installation process is complete, there is some custom configuration you must do. Follow the instructions below

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.

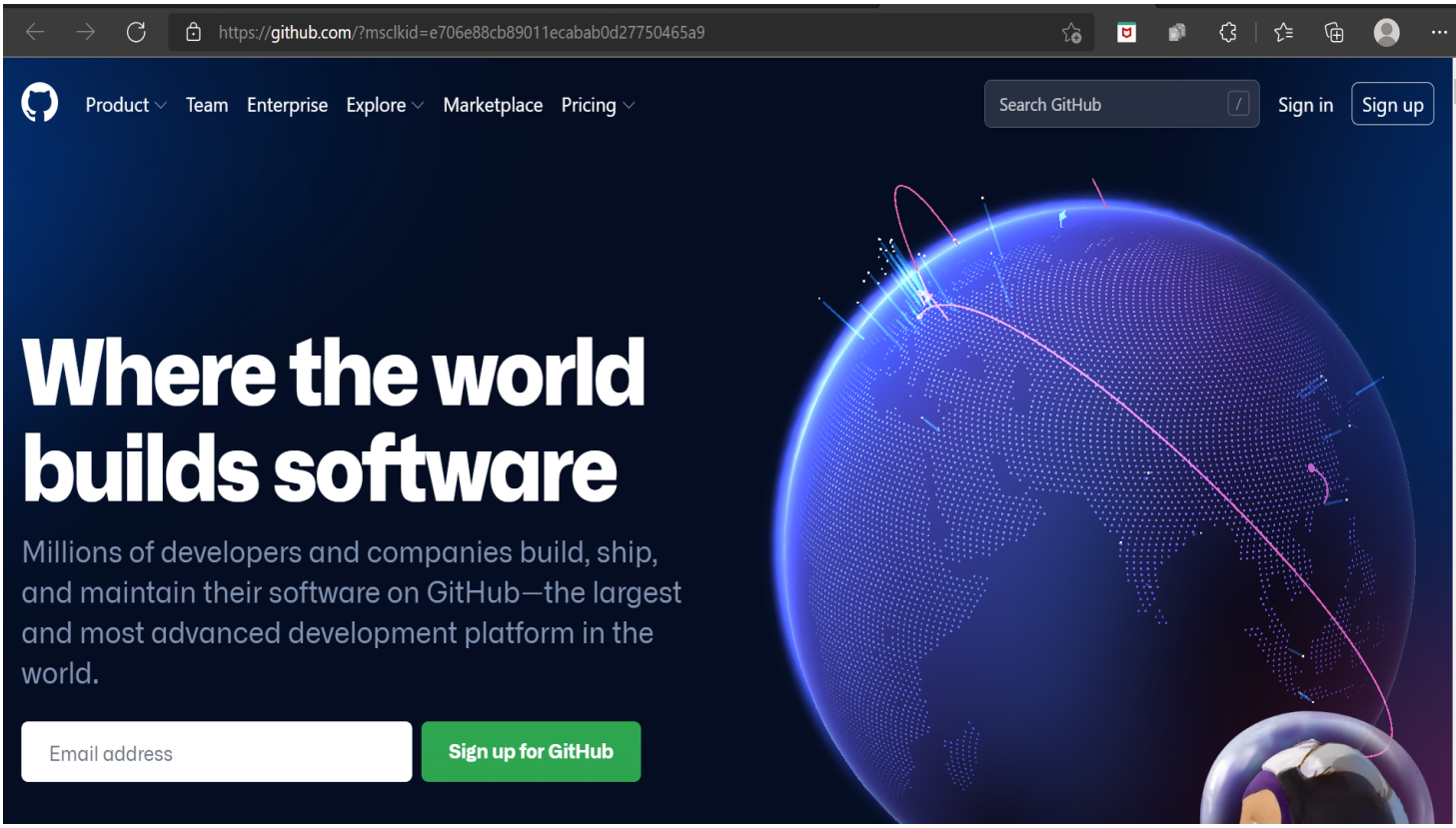


Experiment 2

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.

CHITKARA UNIVERSITY

Experiment 3

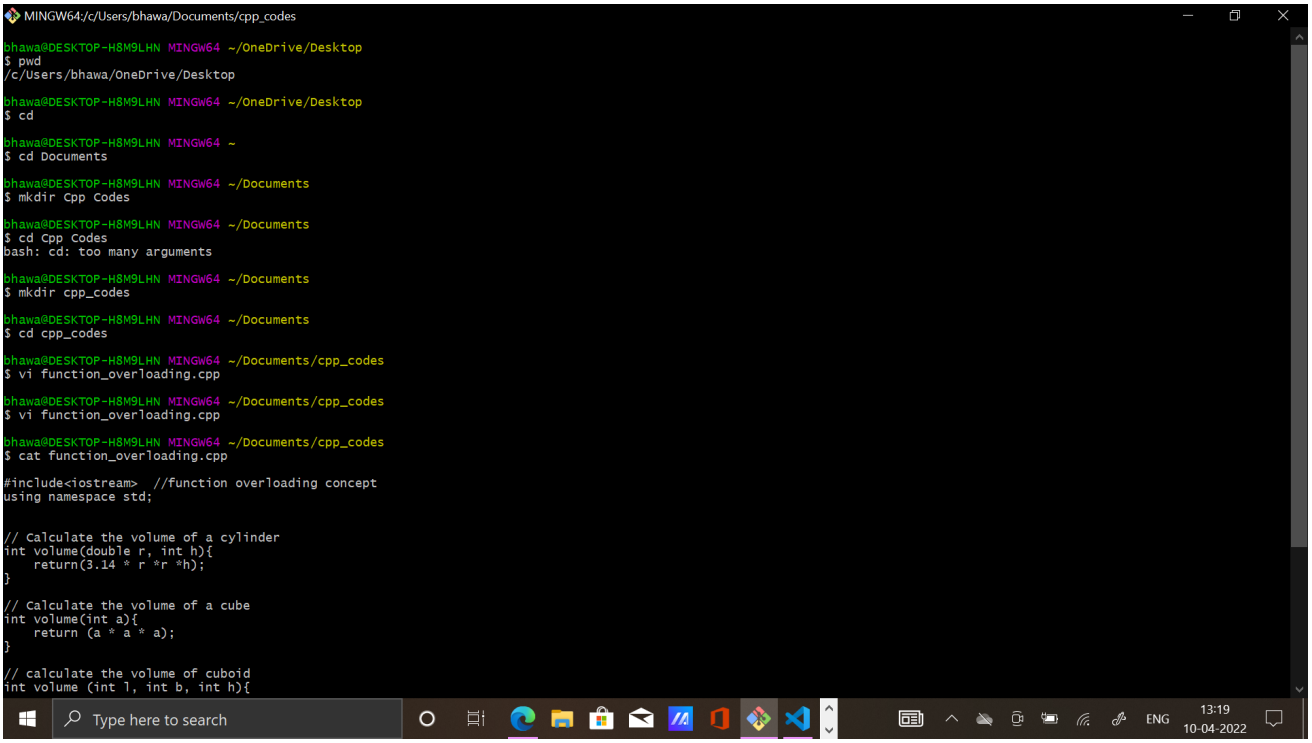
Aim: How to use Git log
learning basic Git commands and how to make a repository.

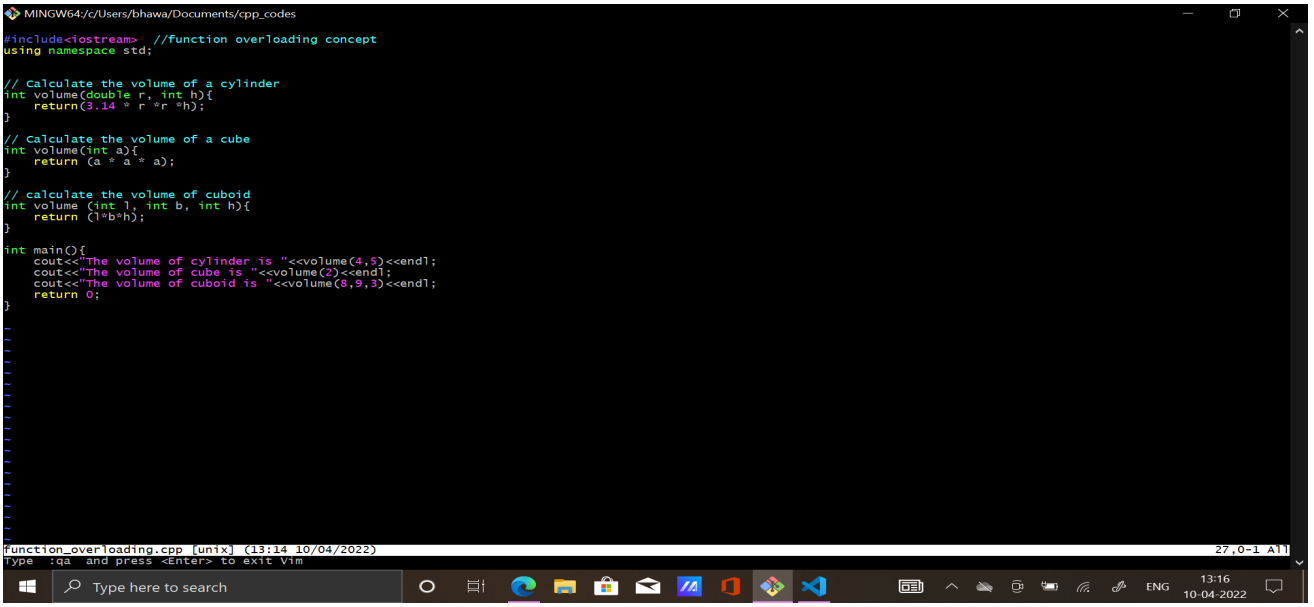
Description: We Will take a look at some basic commands

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

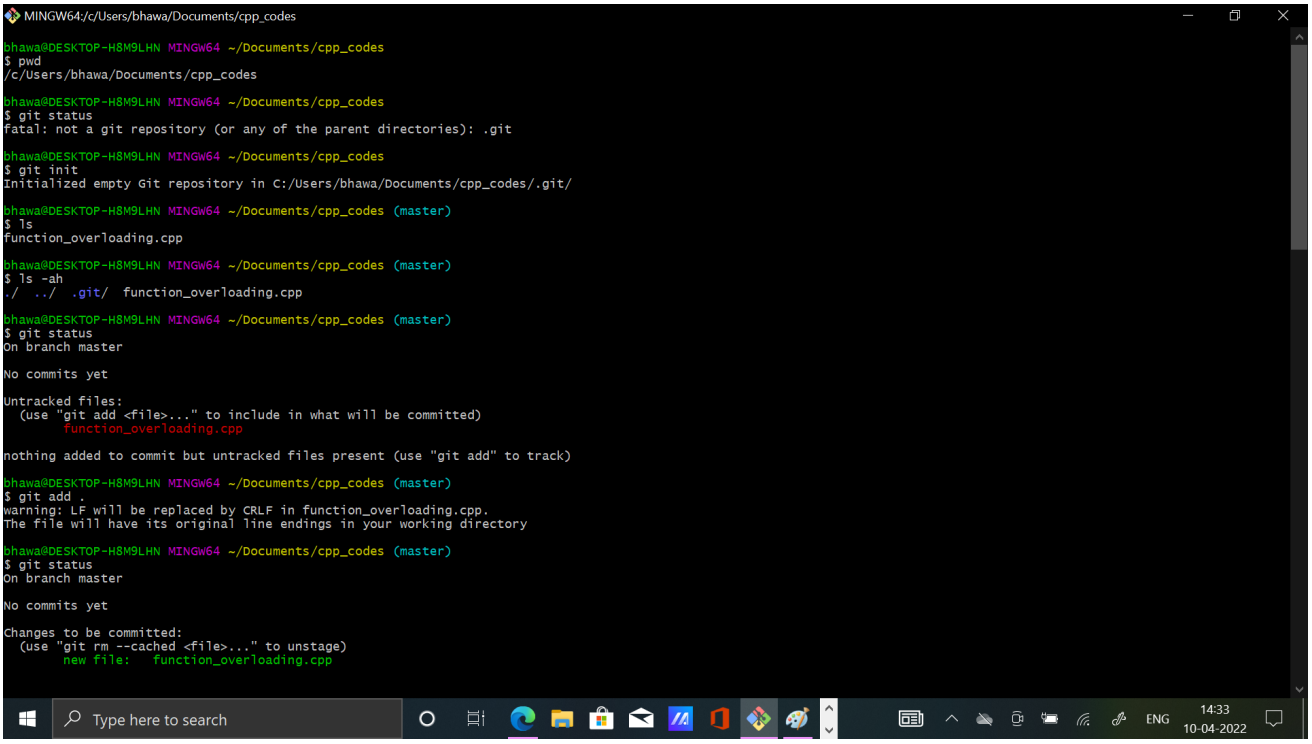
All these commands are shown in the screenshots below you can also take help from there

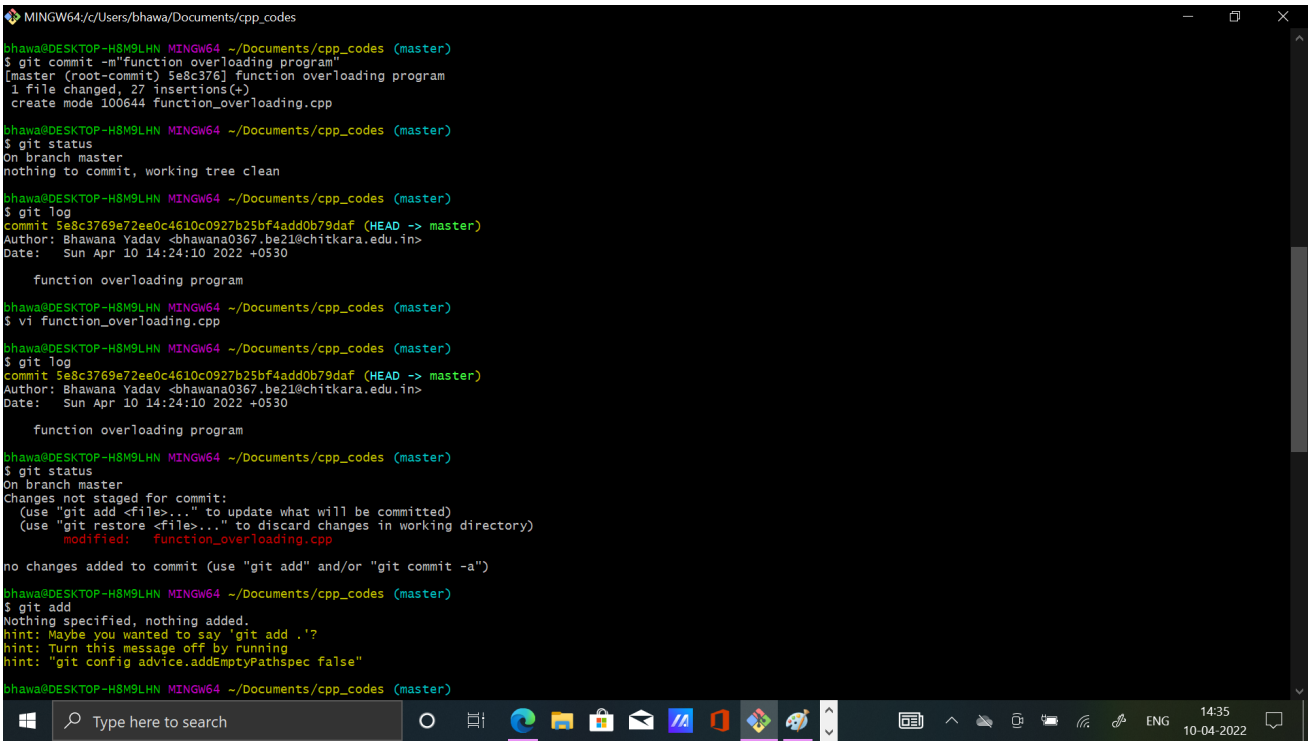


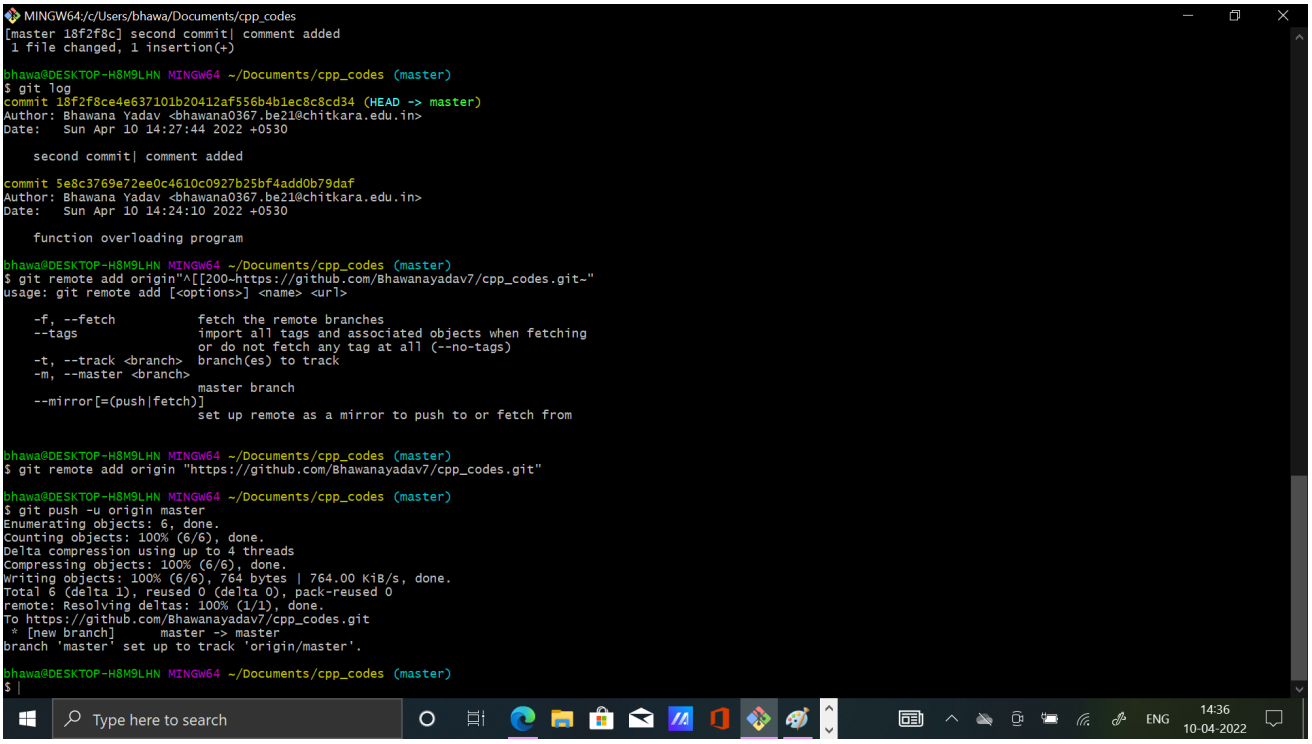


Running Git commands and making a repository

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
Type here to search
ENG
14:33
10-04-2022


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)
Type here to search
ENG
14:35
10-04-2022



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).

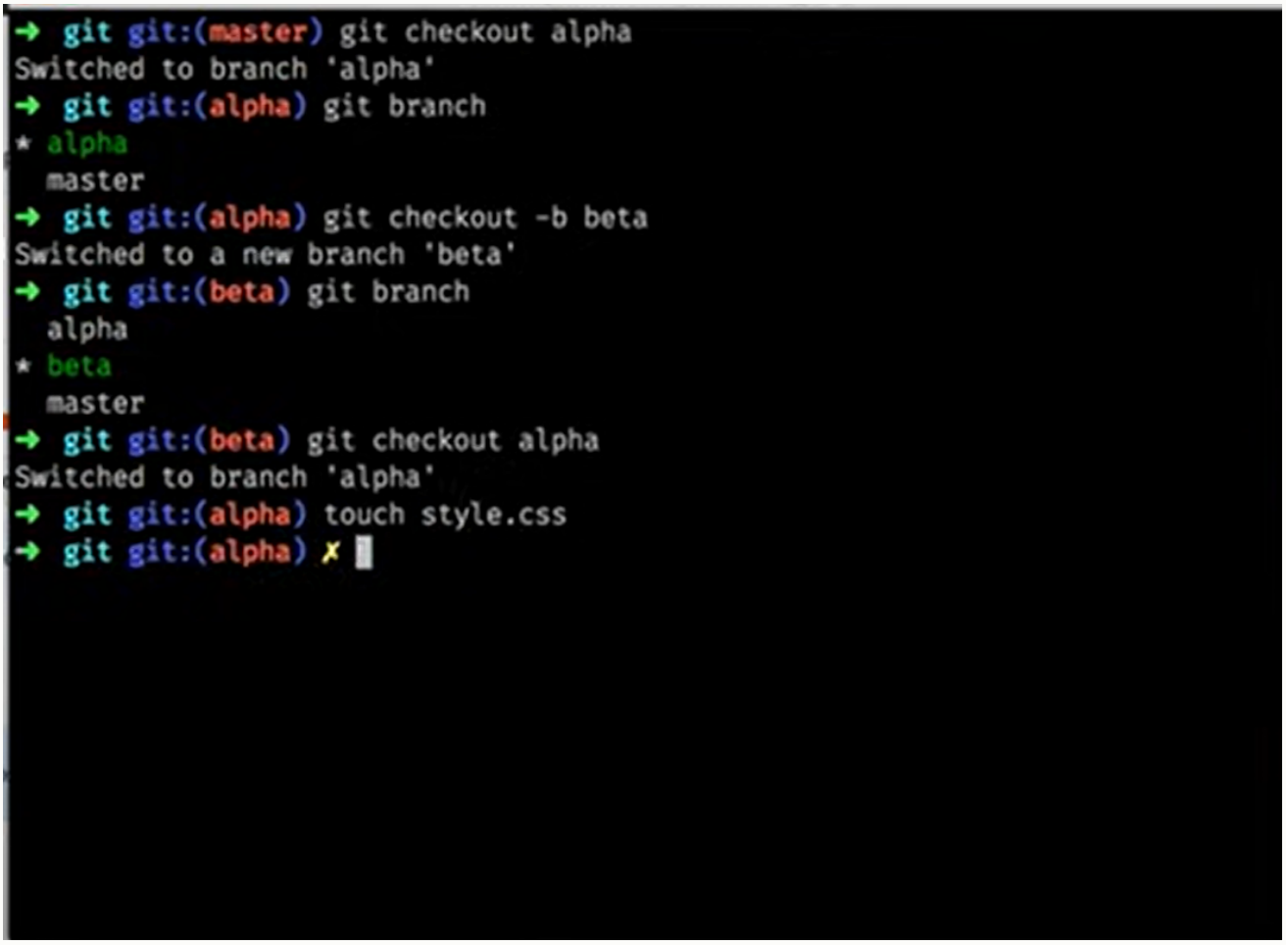
1. To check how many branches we have : git branch

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) ✗

CHITKARA UNIVERSITY

Experiment 5

Aim: Git lifecycle description

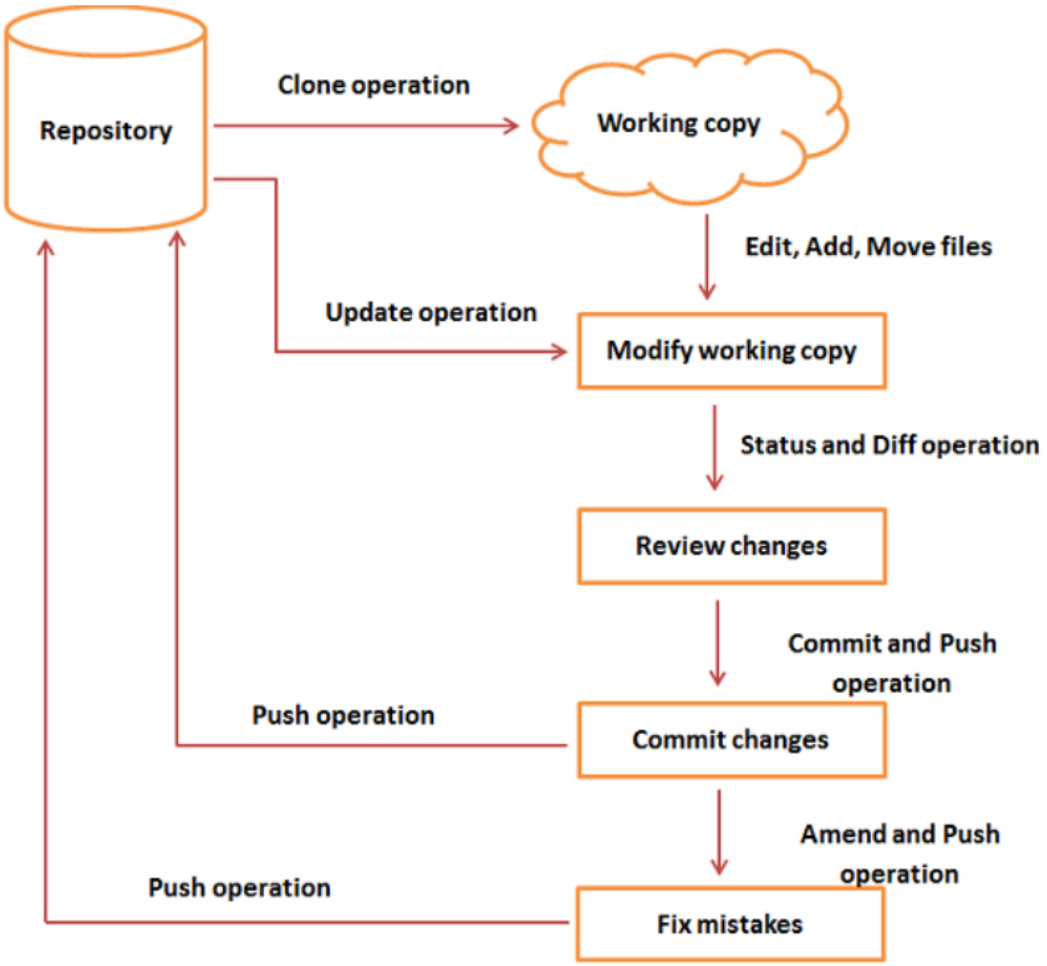
General workflow is as follows −

- 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.

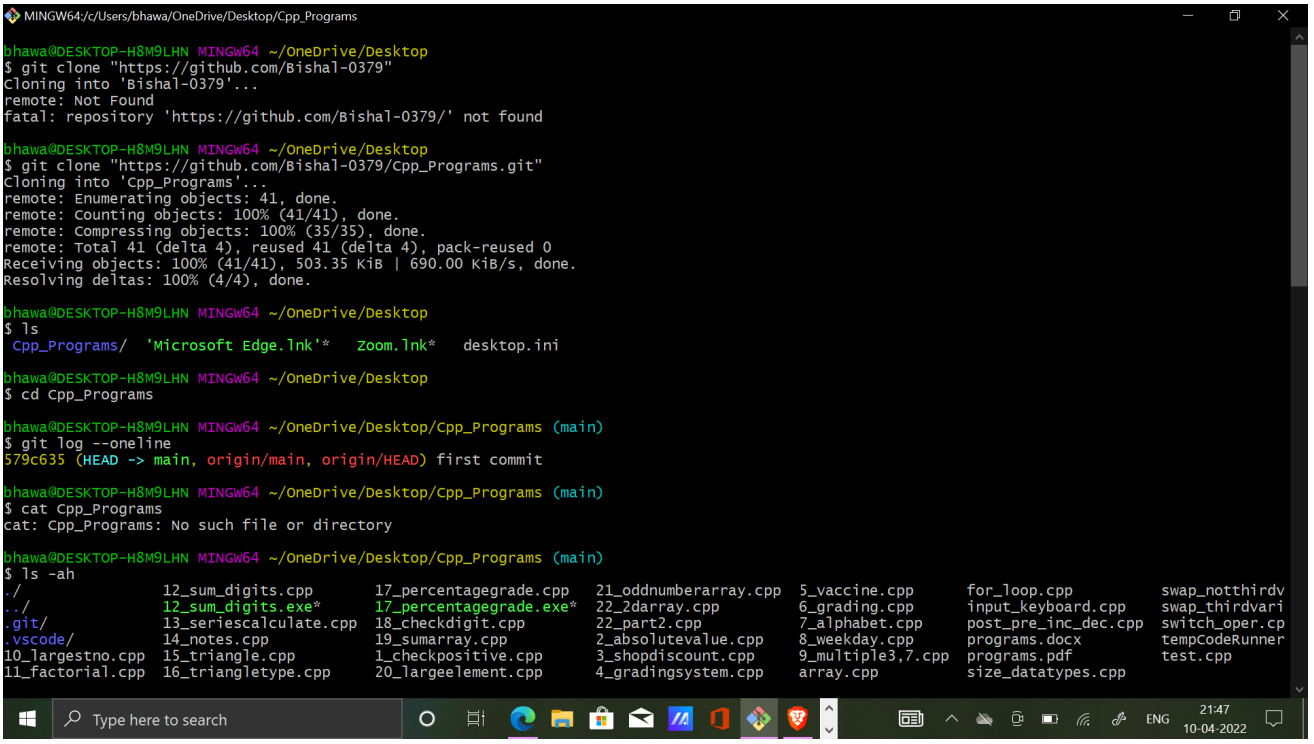
CHITKARA UNIVERSITY

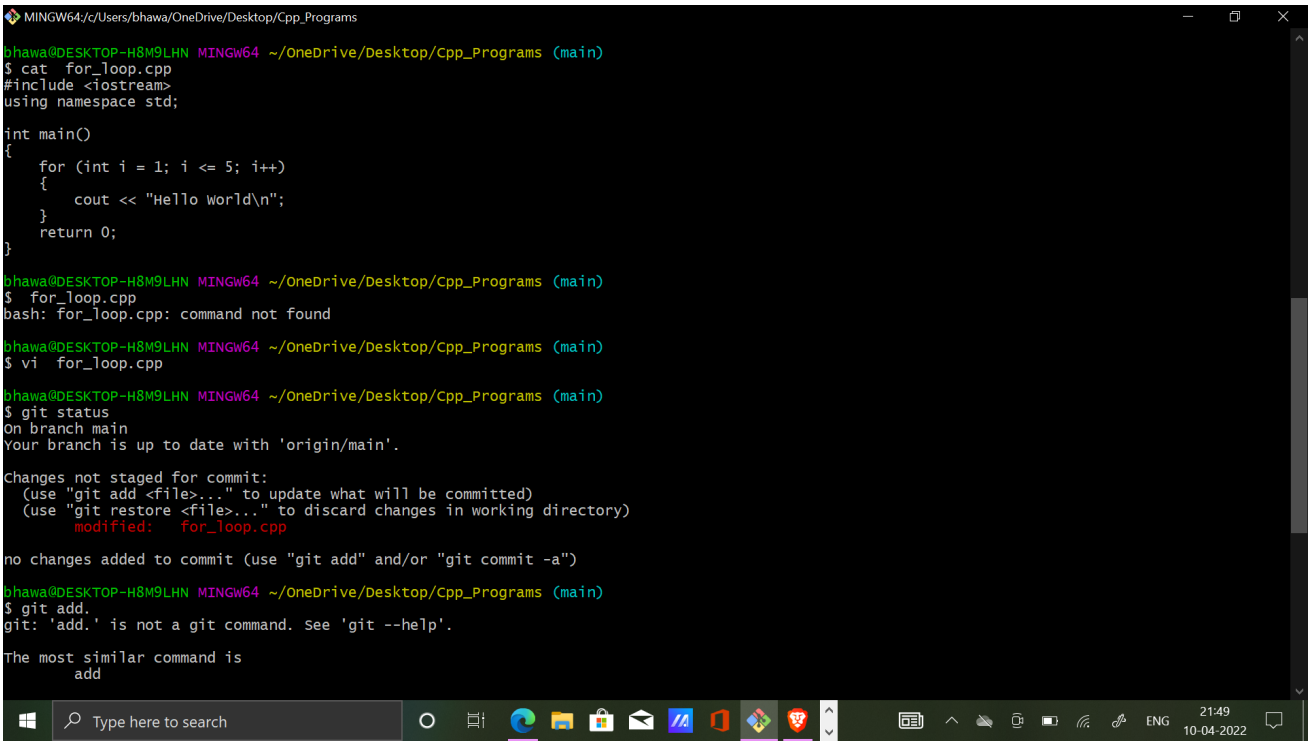
Work-flow

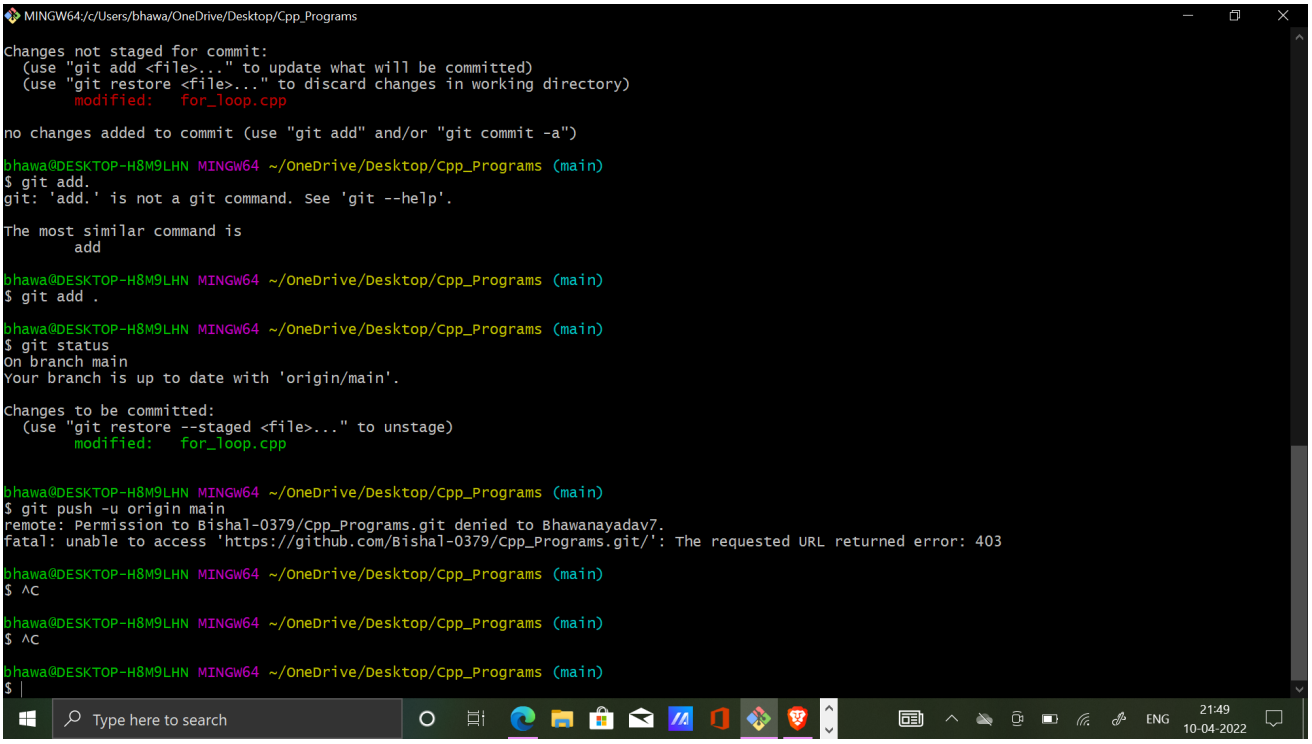
Shown below is the pictorial representation of the work-flow.



CHITKARA UNIVERSITY

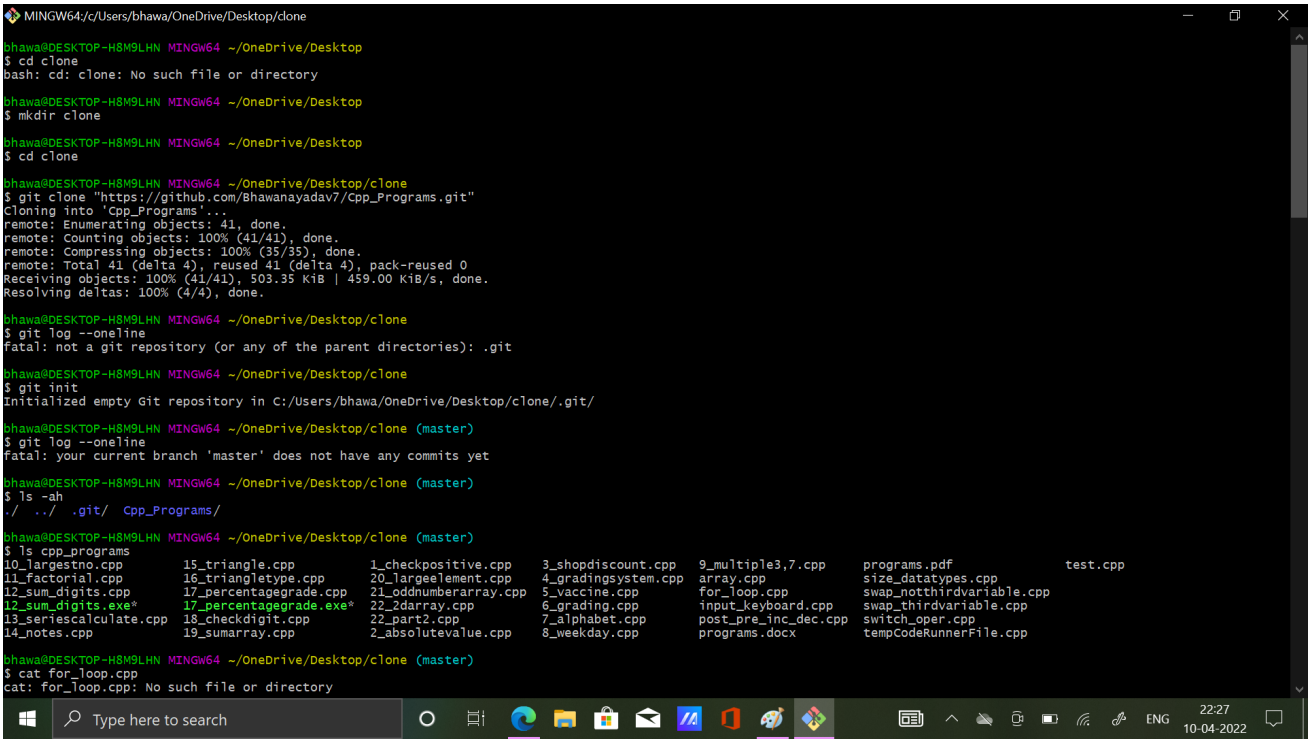


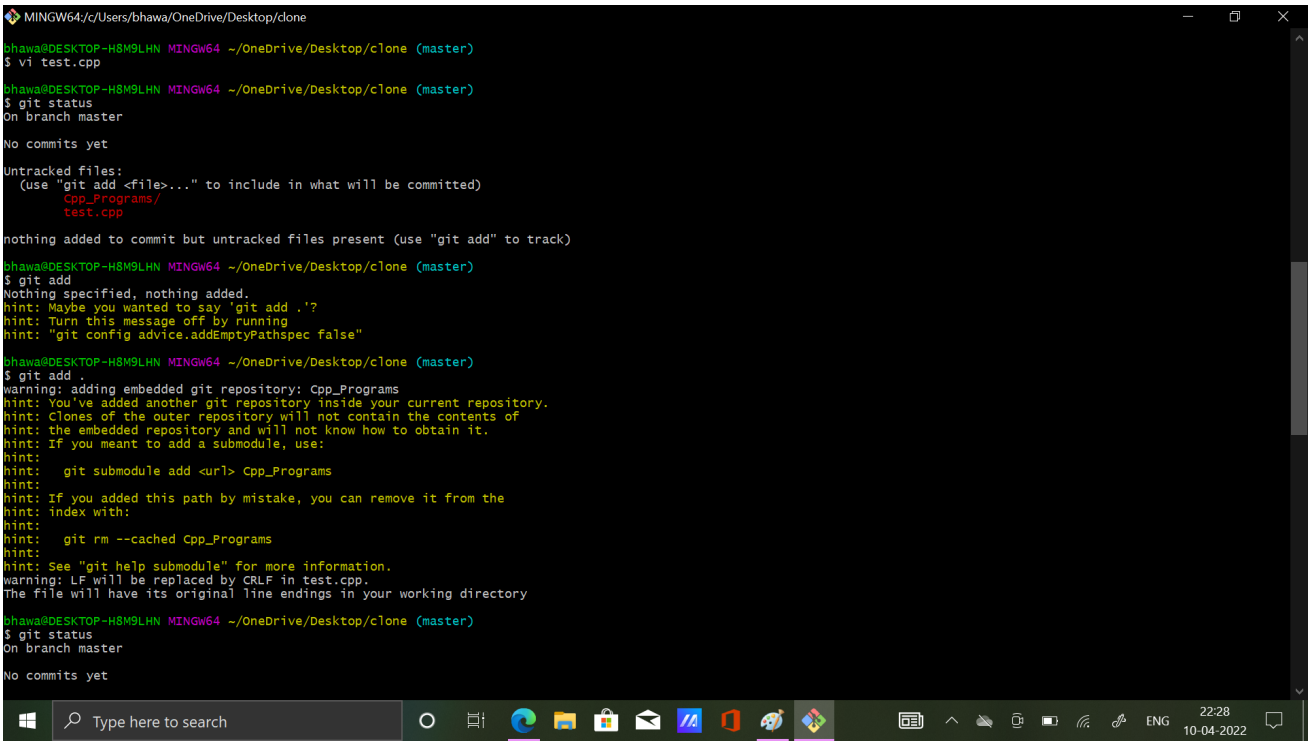


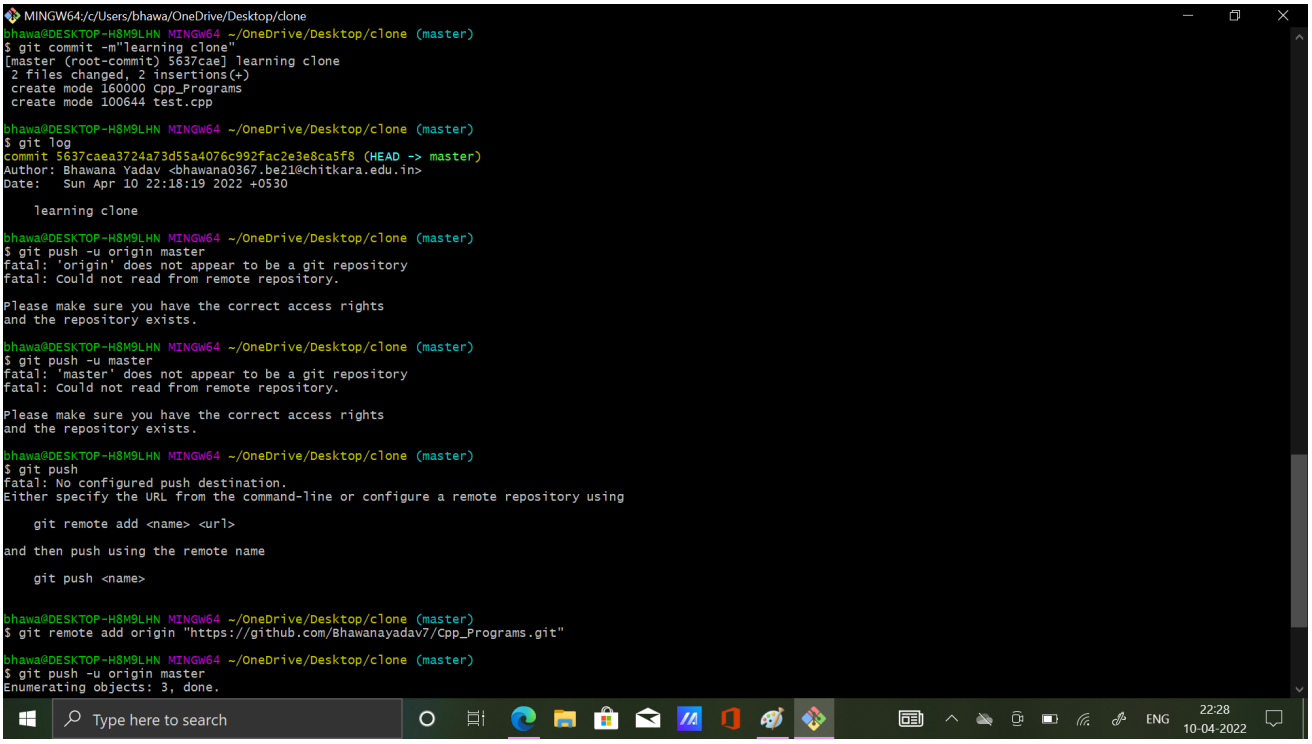


so it showed permission denied which is because we did not forked it and tried to make changes in some else repository which is not allowed.

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: clone: 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_largeelement.cpp 4_gradingsystem.cpp array.cpp size_datatypes.cpp
12_sum_digits.cpp 17_percentagegrade.cpp 21_oddnumberarray.cpp 5_vaccine.cpp for_loop.cpp swap_notthirdvariable.cpp
12_sum_digits.exe* 17_percentagegrade.exe* 22_2darray.cpp 6_grading.cpp input_keyboard.cpp swap_thirdvariable.cpp
13_seriescalculate.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
Type here to search
ENG
22:27
10-04-2022


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:
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:
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
Type here to search
ENG
22:28
10-04-2022



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

Task 1.2

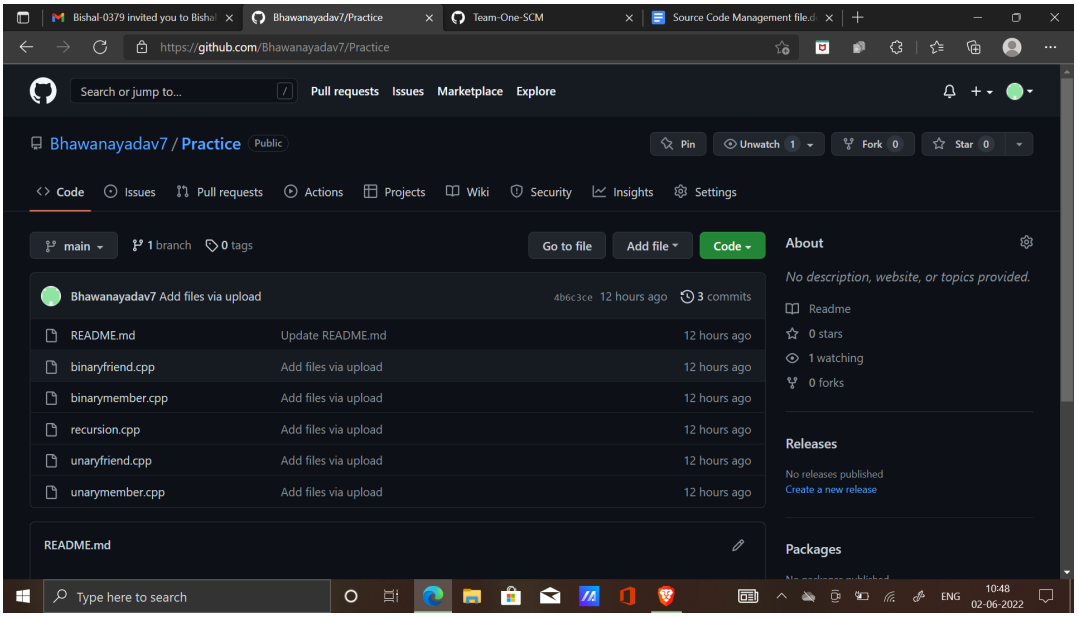
Aim : Add collaborators on GitHub.

Why add collaborators?

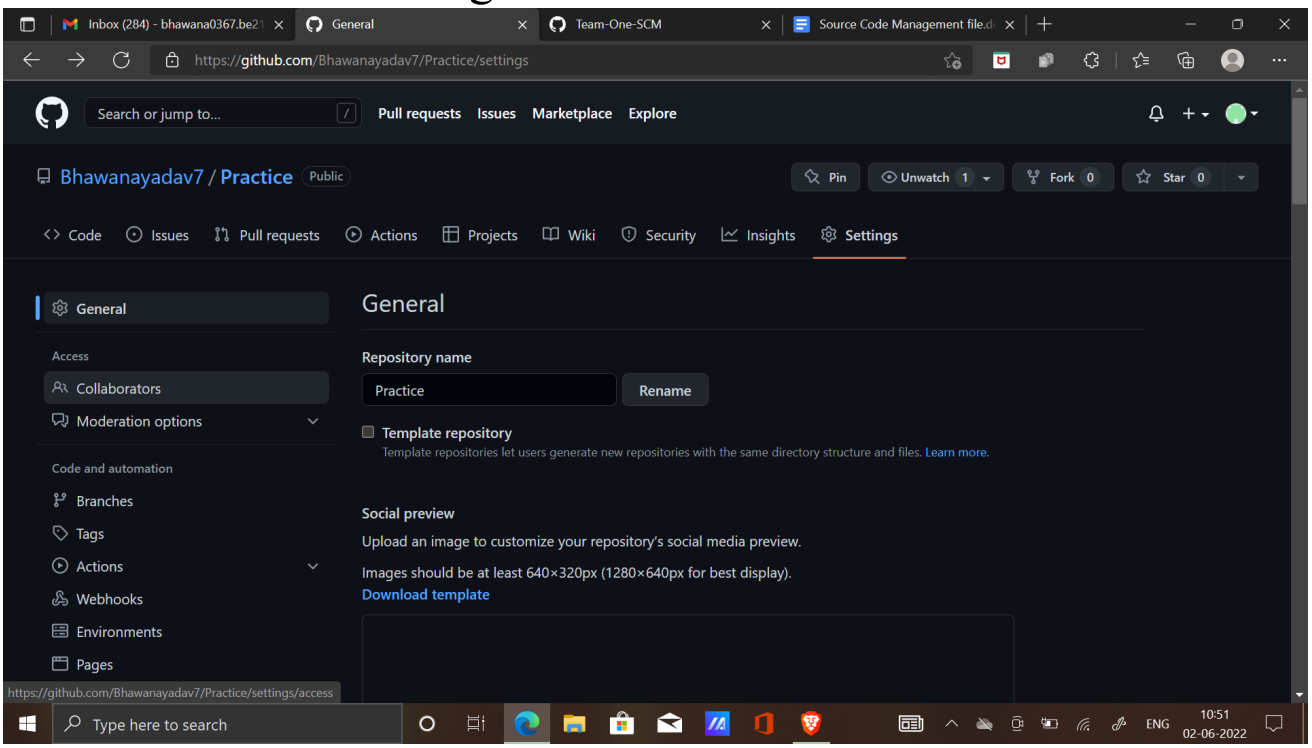
In GitHub, we can invite other GitHub users to become collaborators to our private repositories (which expires after 7 days if not accepted, restoring any unclaimed licenses). Being a collaborator of a personal repository you can pull (read) the contents of the repository and push (write) changes to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

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

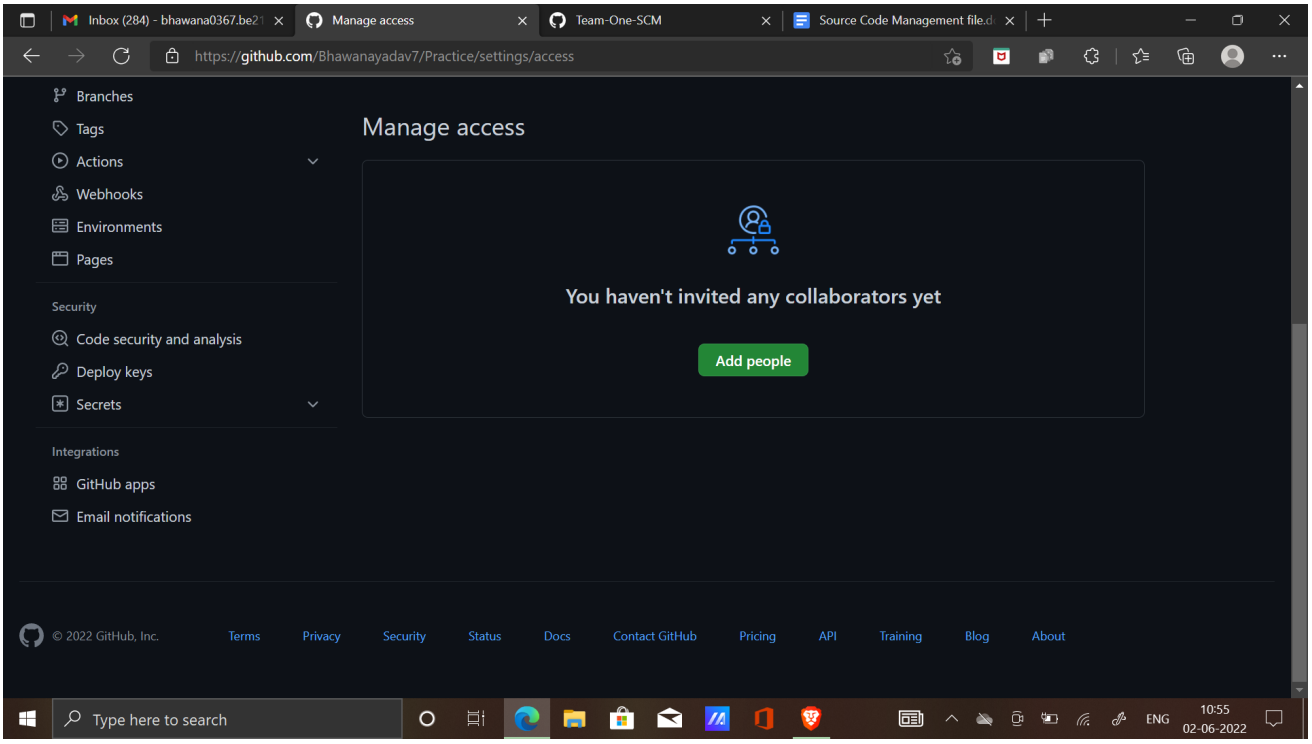
Step 1 : Click on the repository on which you want to add the collaborators then go to the Settings tab in the right corner of the GitHub page.



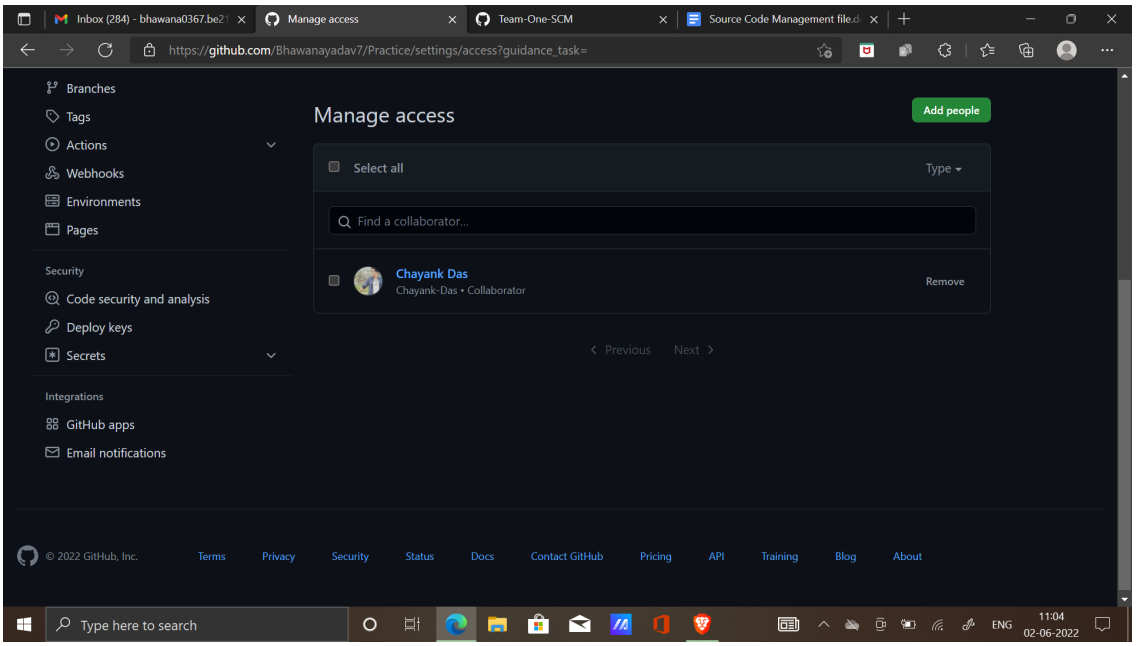
Step 2 : Go to the Collaborators option under the Settings tab. On the Collaborators page, you will see an Add people link as shown in the below diagram.



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.



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.

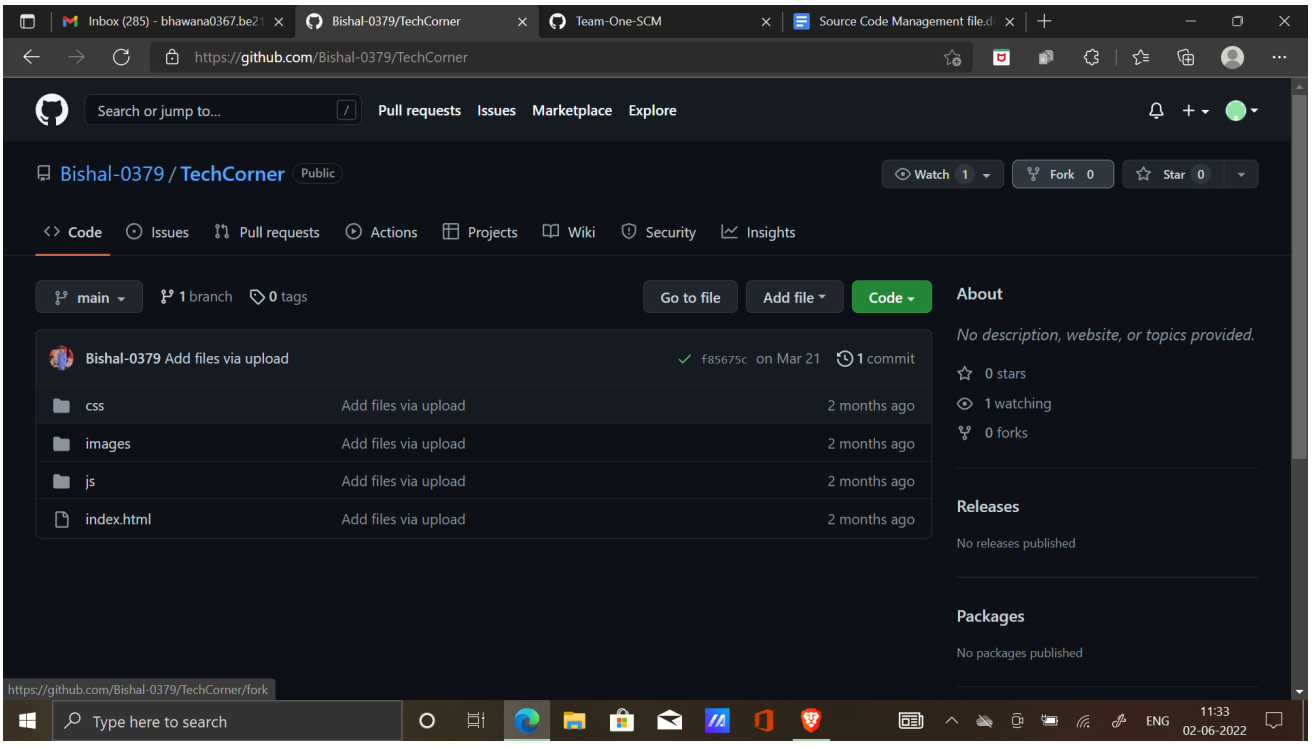


Aim: Fork And Commit

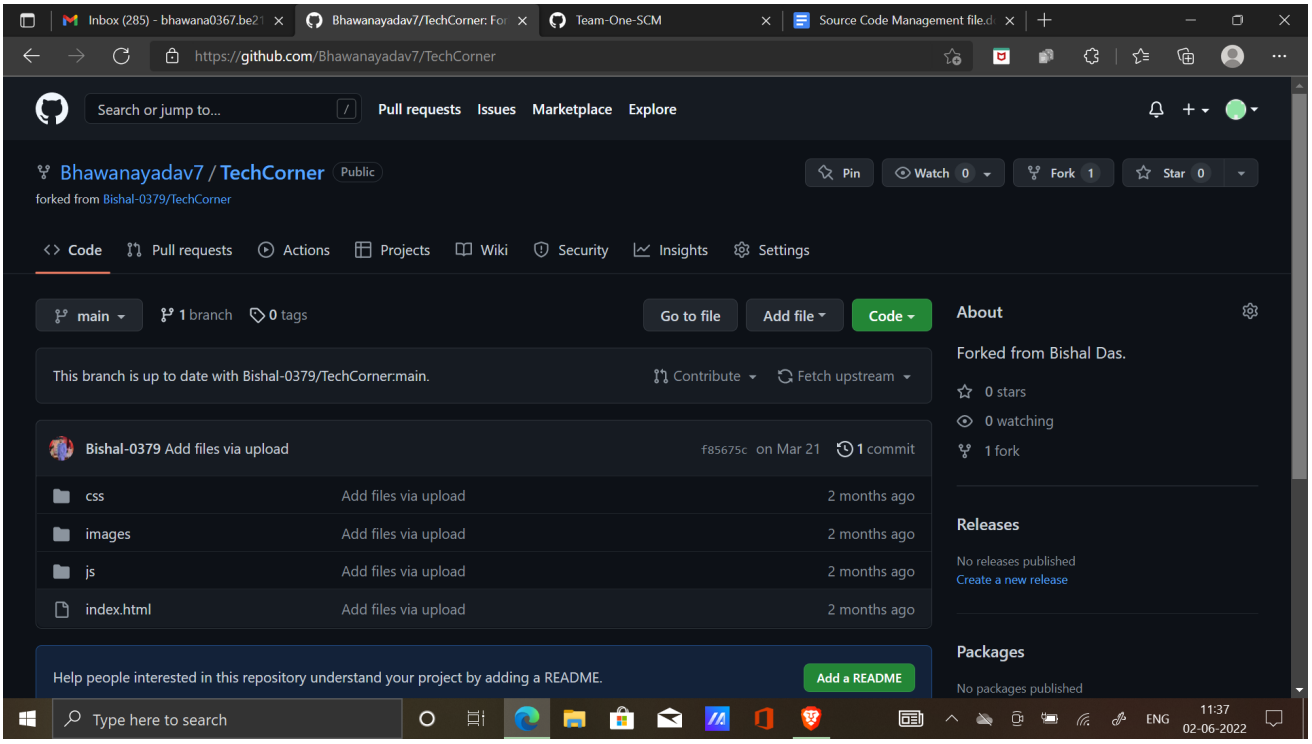
Fork is a copy of an original repository in which you can make changes without affecting the original repository.

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

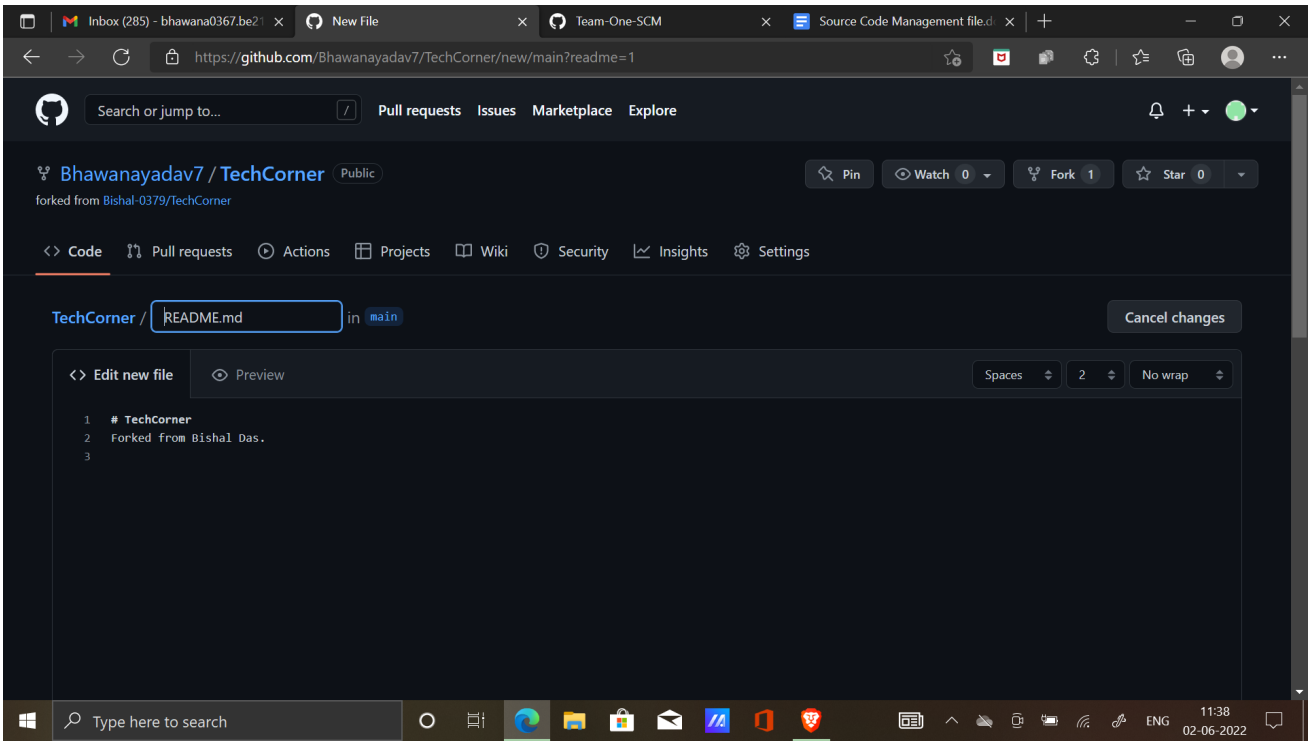
Step 1 : To fork a repository, Click on the Fork button as shown below



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

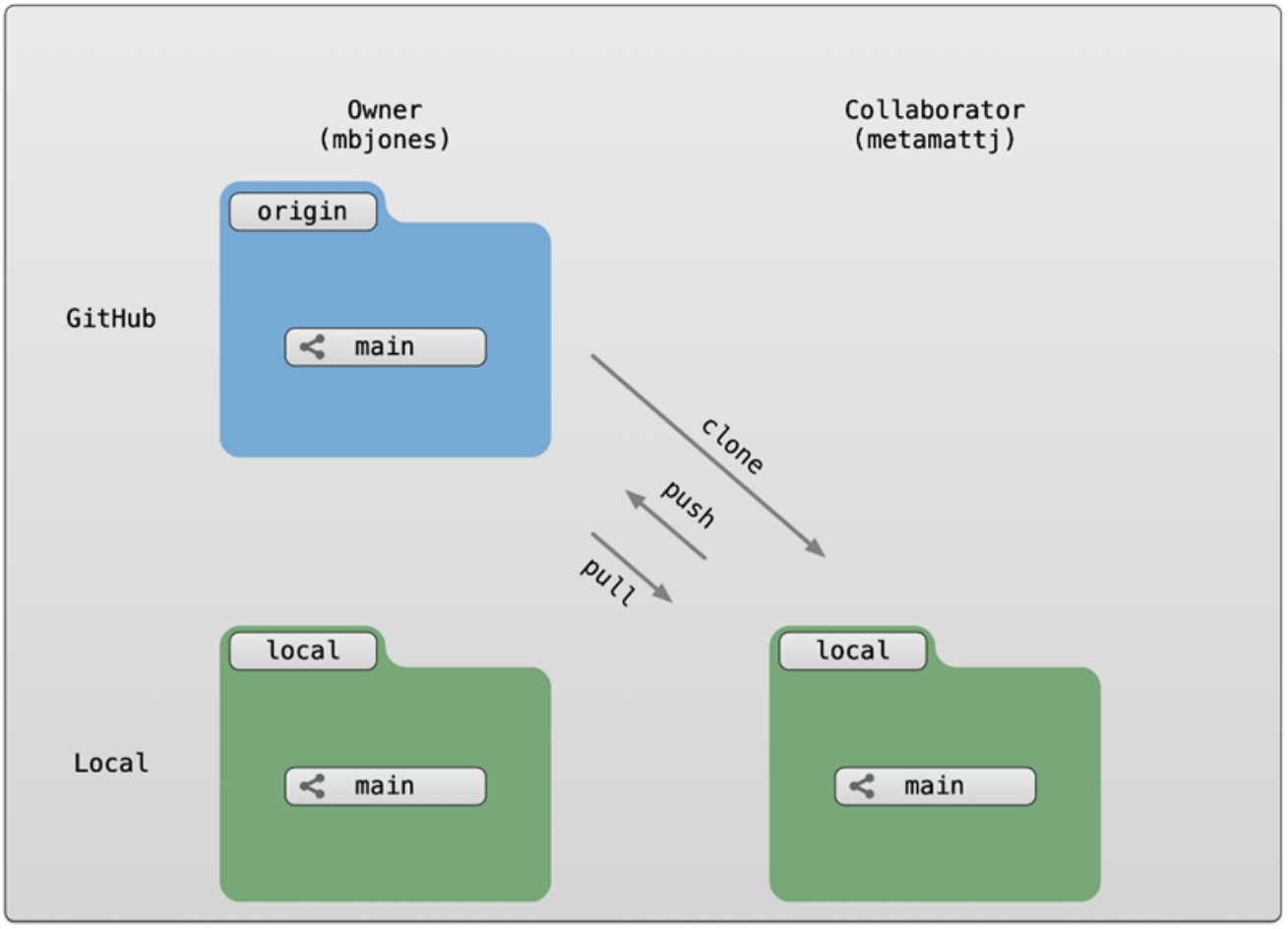


Step 3 : After the forking process is complete, you can commit changes in the file.



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.

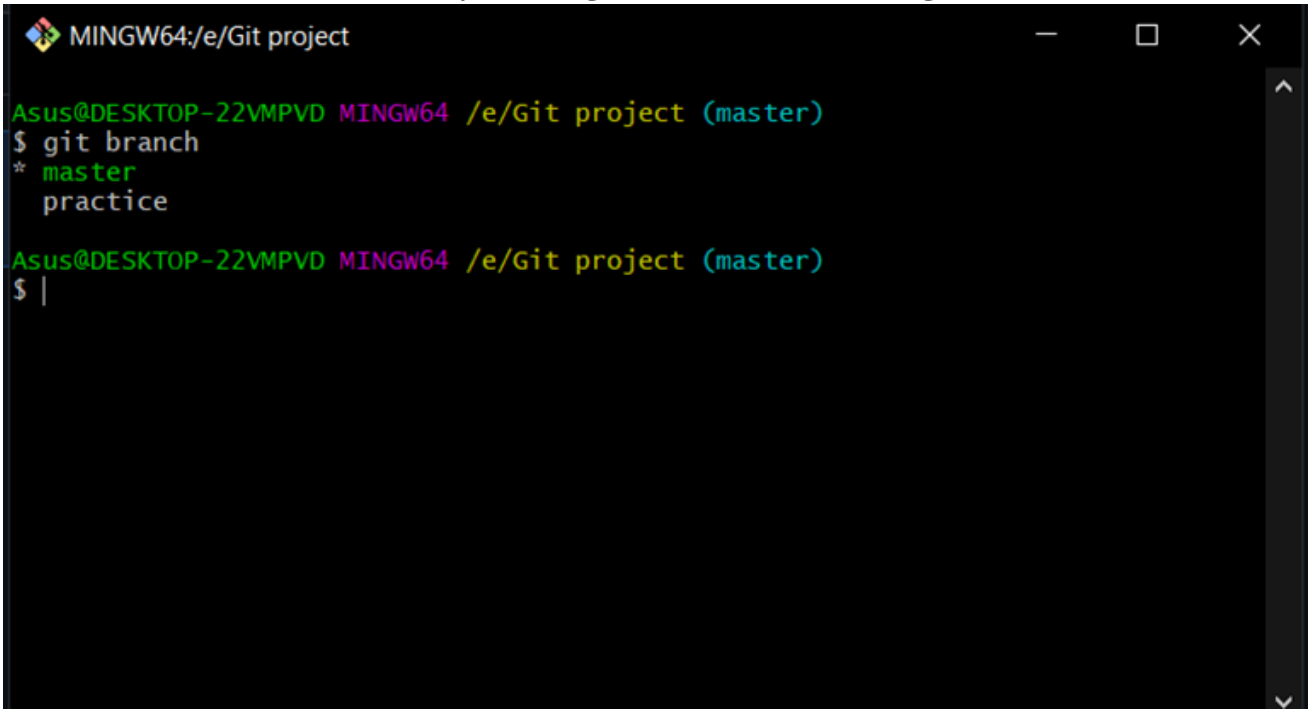


Git Merge

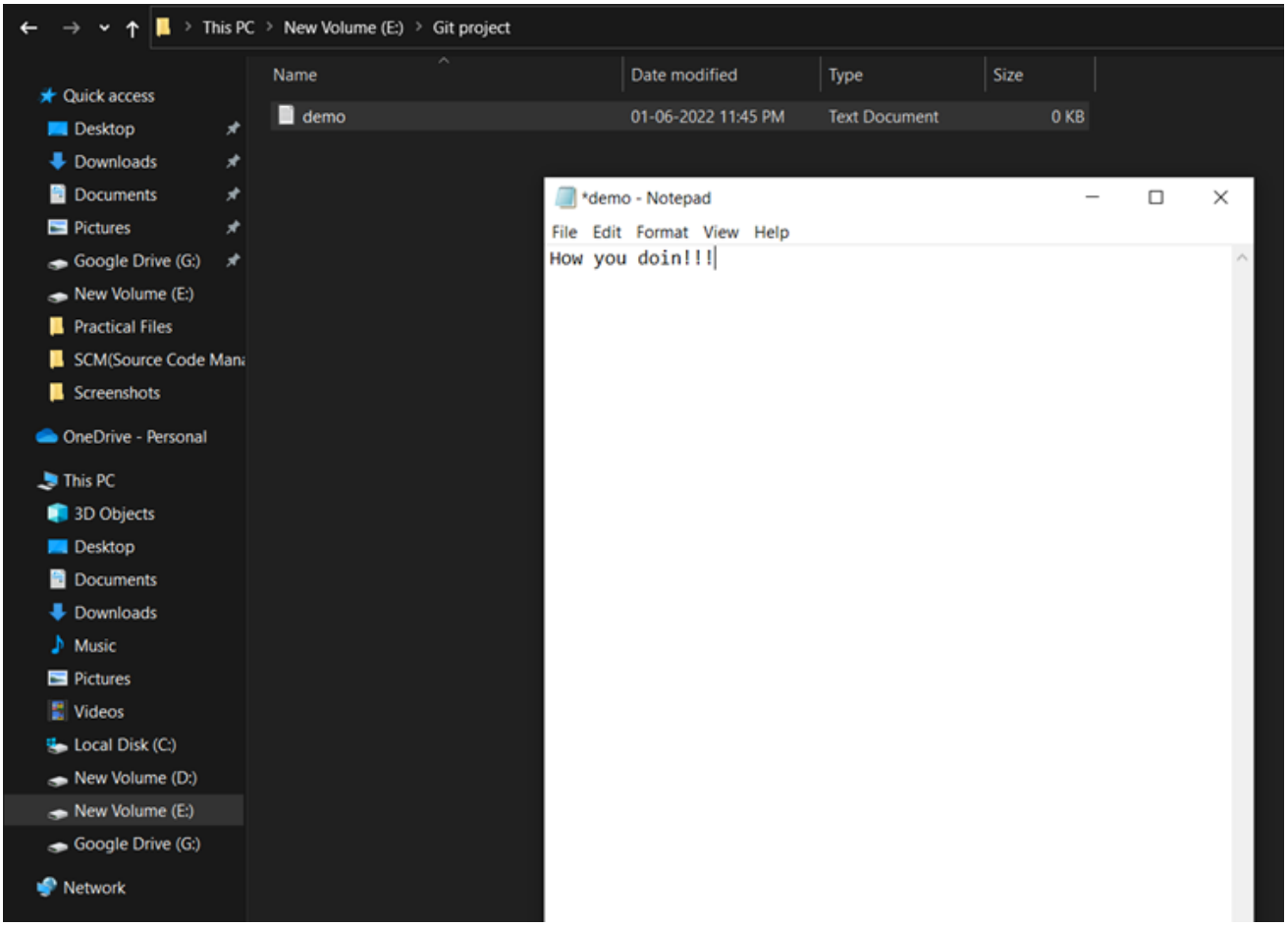
Git merge will combine multiple sequences of commits into one

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 :-

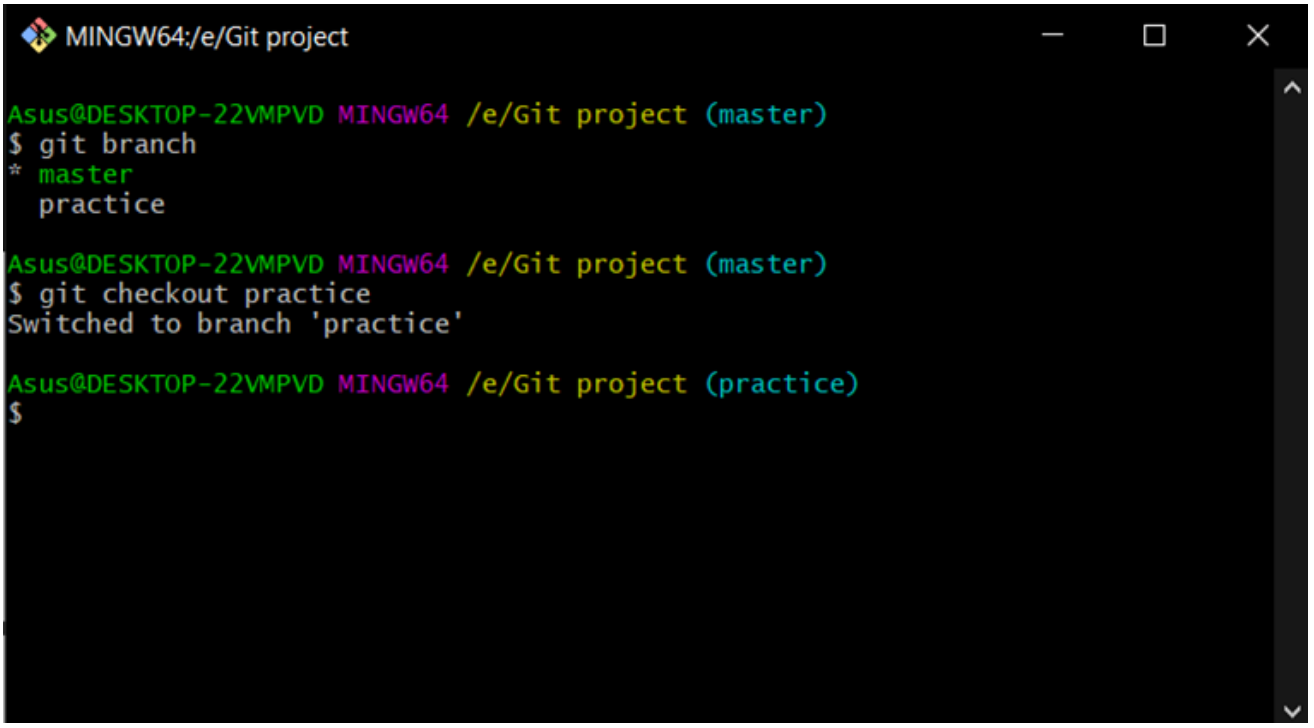
Step 1 - Below are the branches that are created in my repo. We can see our branches by using this command git branch.

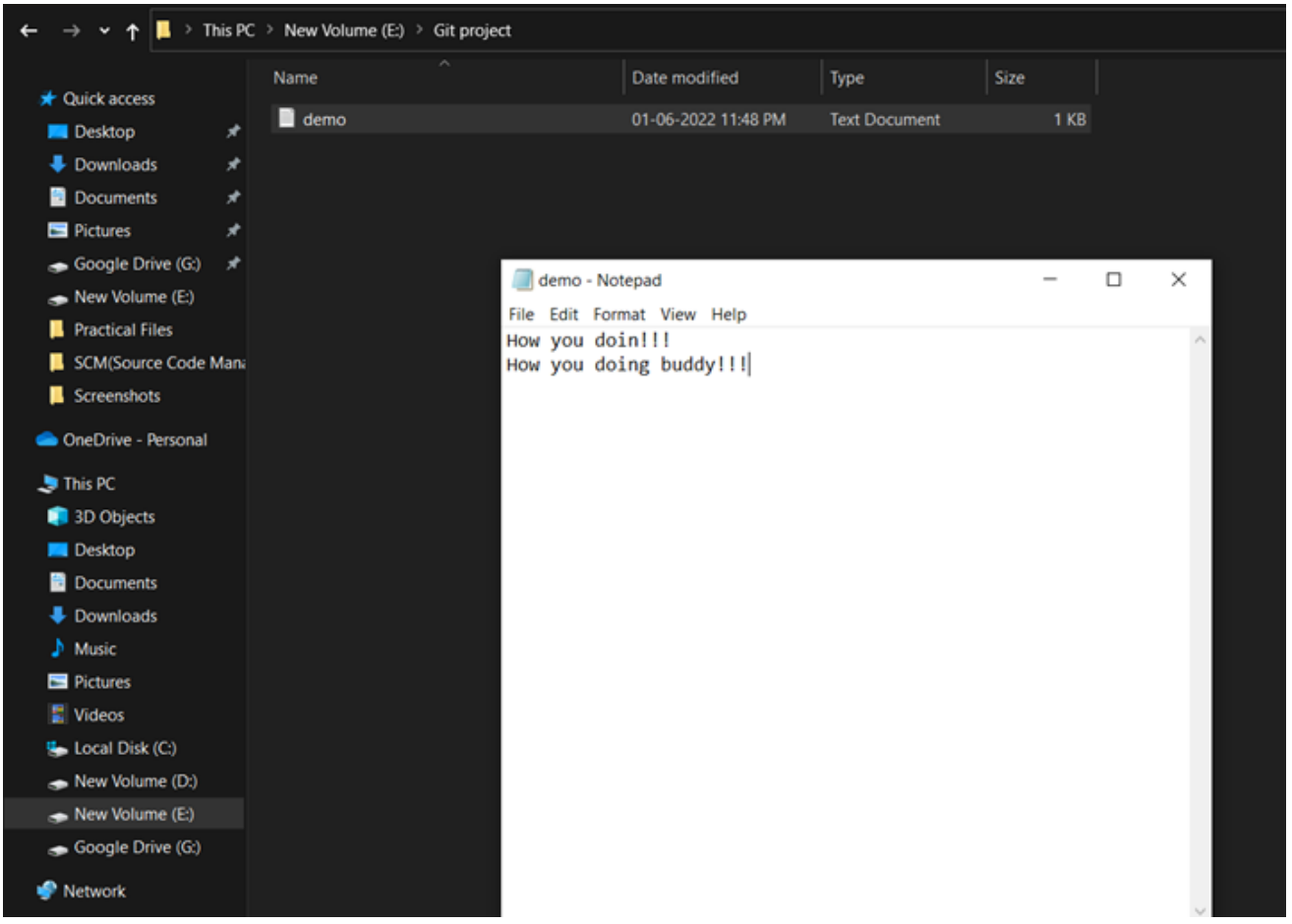


Step 2 – Created a new text file called new demo.txt in the master branch and inserted text ‘how you doin !!!’ and saved it.



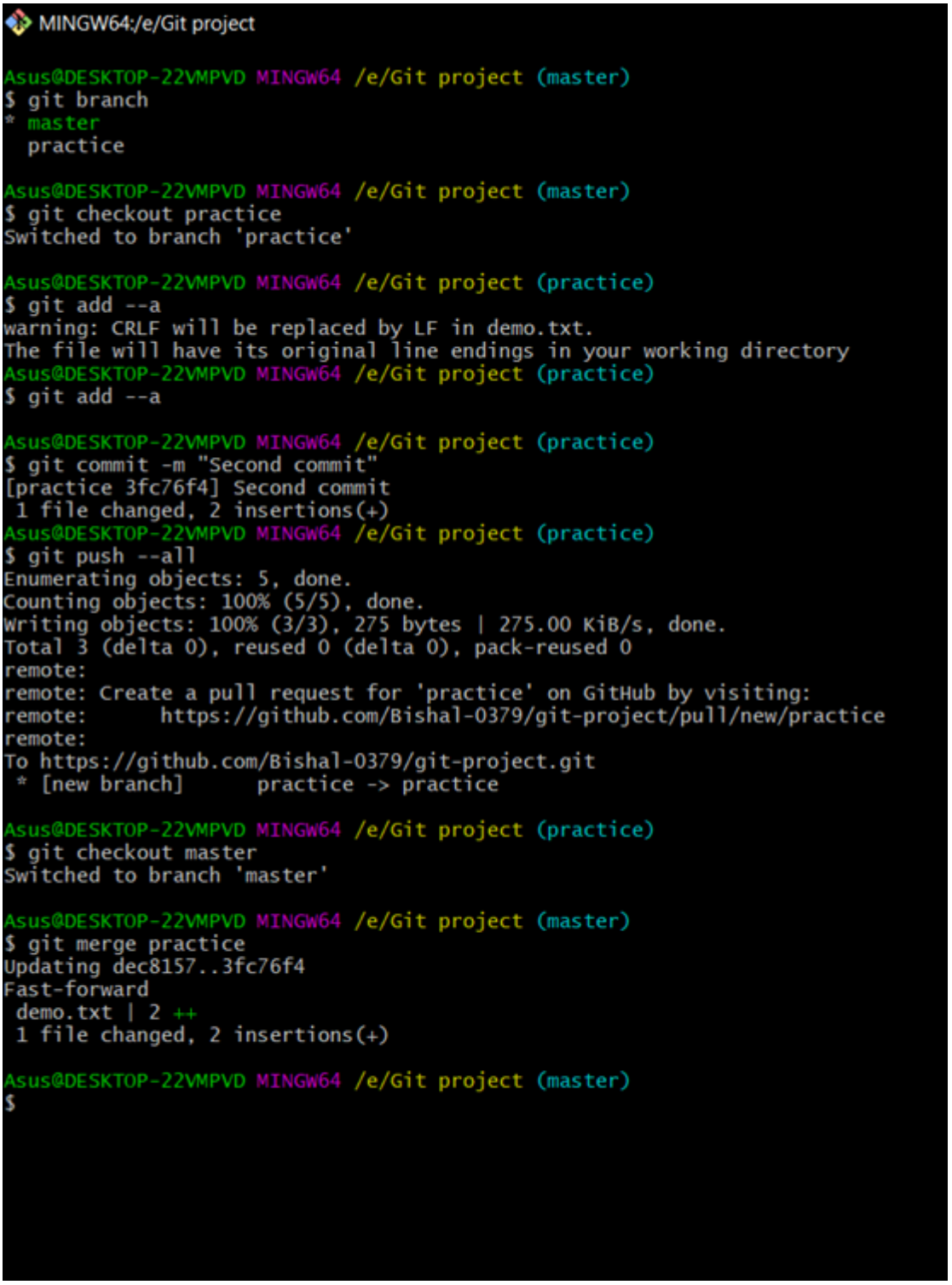
Step 3 – Switch to another branch that we already created called practice by using the command git checkout (branch name) and added new text ‘how you doing buddy !!! ’ in the new demo.txt files


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)
$


This PC
New Volume (E:)
Git project
Quick access
Desktop
Downloads
Documents
Pictures
Google Drive (G:)
New Volume (E:)
Practical Files
SCM(Source Code Mana
Screenshots
OneDrive - Personal
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Local Disk (C:)
New Volume (D:)
New Volume (E:)
Google Drive (G:)
Network
Name
Date modified
Type
Size
demo
01-06-2022 11:48 PM
Text Document
1 KB
demo - Notepad
File Edit Format View Help
How you doin!!!
How you doing buddy!!!

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.

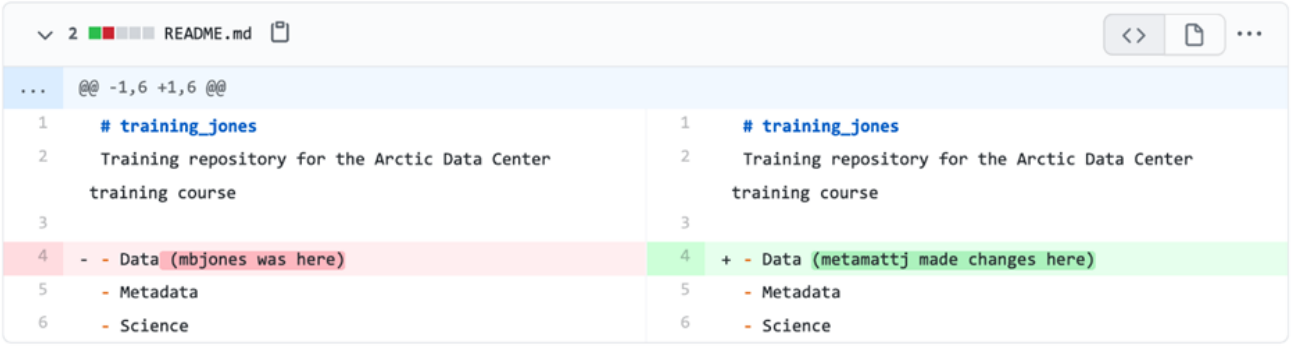
CHITKARA UNIVERSITY



CHITKARA UNIVERSITY

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 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 stull 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**

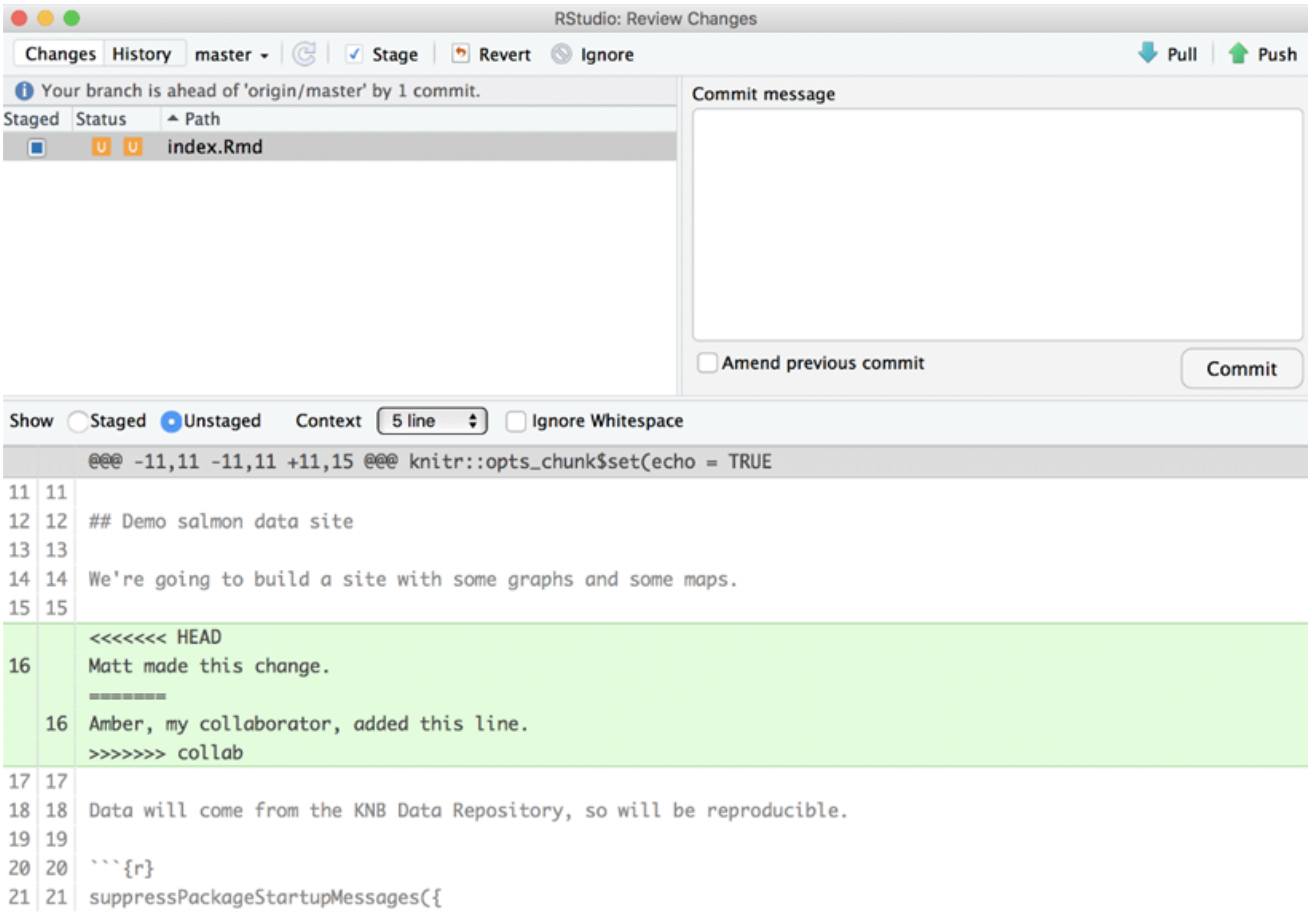
The simplest way to resolve a conflict, given that you know whose version of the file you want to keep, is to use the command line git program to tell git to use either your changes (the person doing the merge), or their changes (the other collaborator).

- 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 you 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:



To resolve the conficts, 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

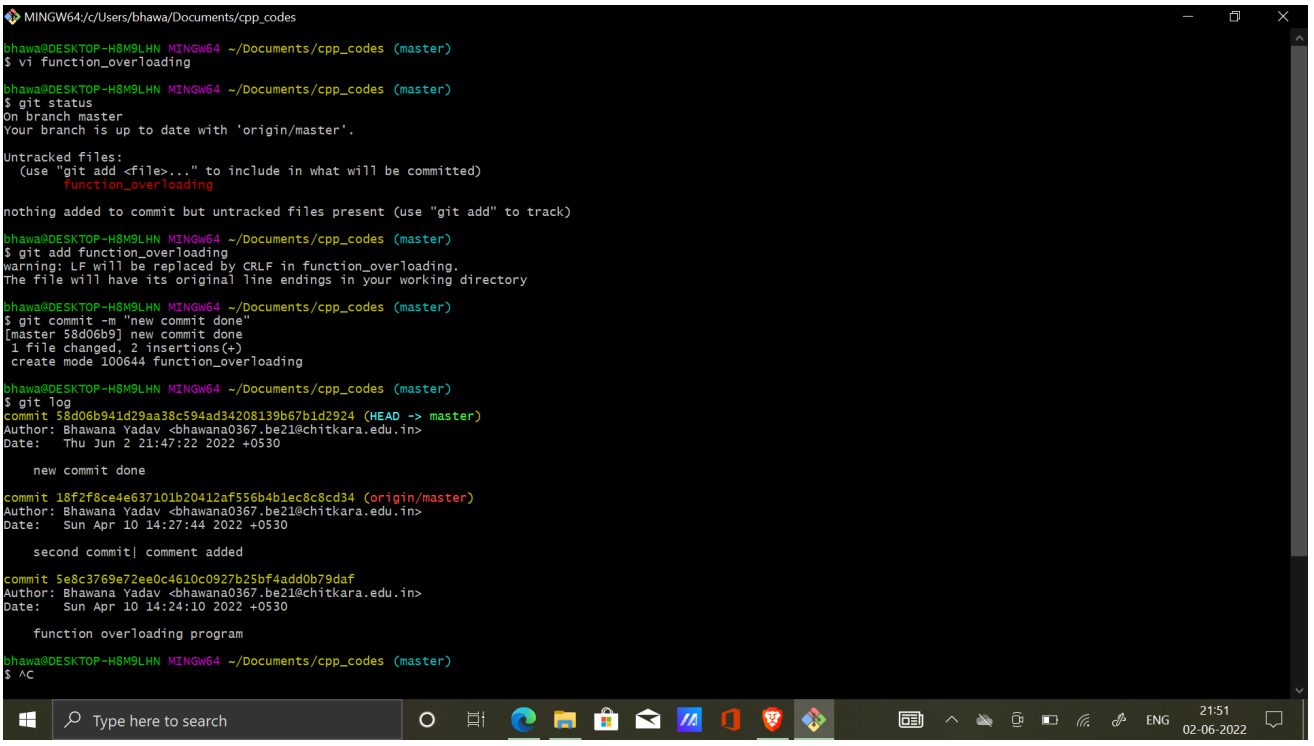
Always start your working sessions with a pull to get any outstanding changes, then start doing your editing and work. Stage your changes, but before you commit, Pull again to see if any new changes have arrived. If so, they should merge in easily if you are working in different parts of the program. 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.

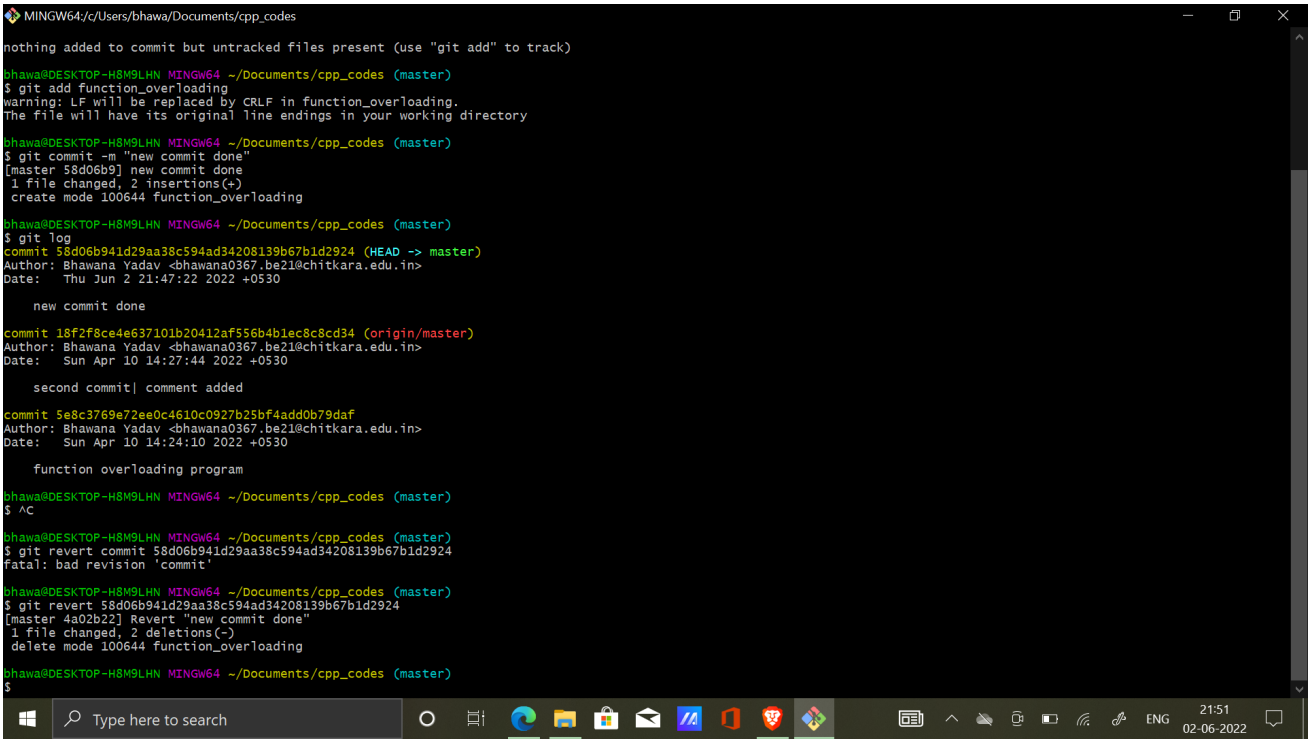
Aim: Reset and Revert

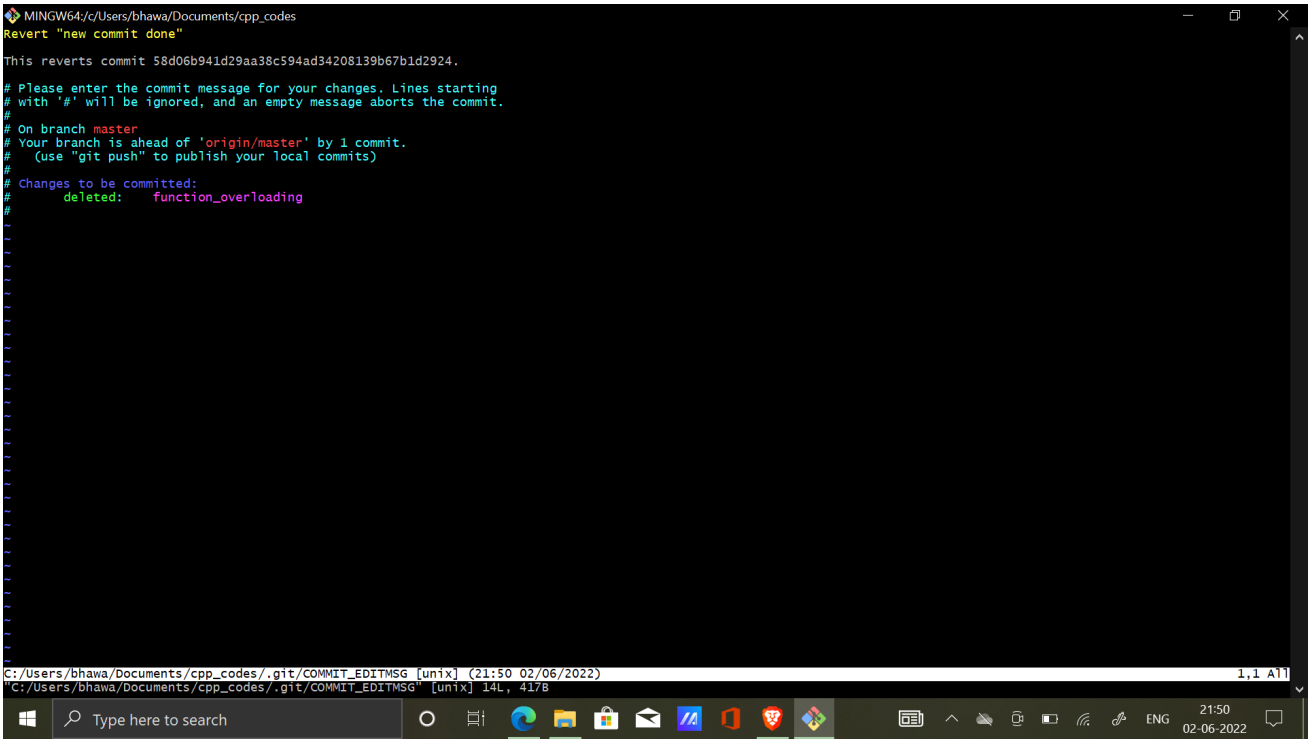
What is git revert?

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.

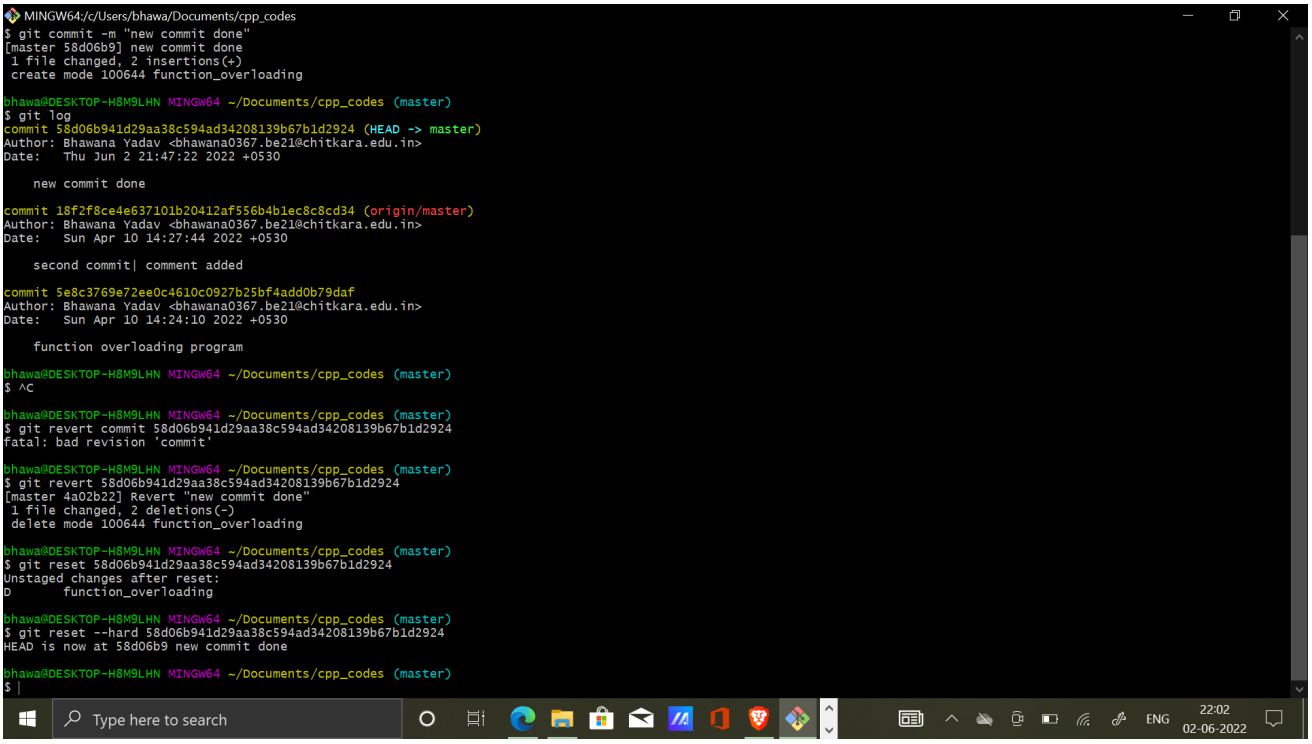






What is git reset?

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



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

CHITKARA UNIVERSITY

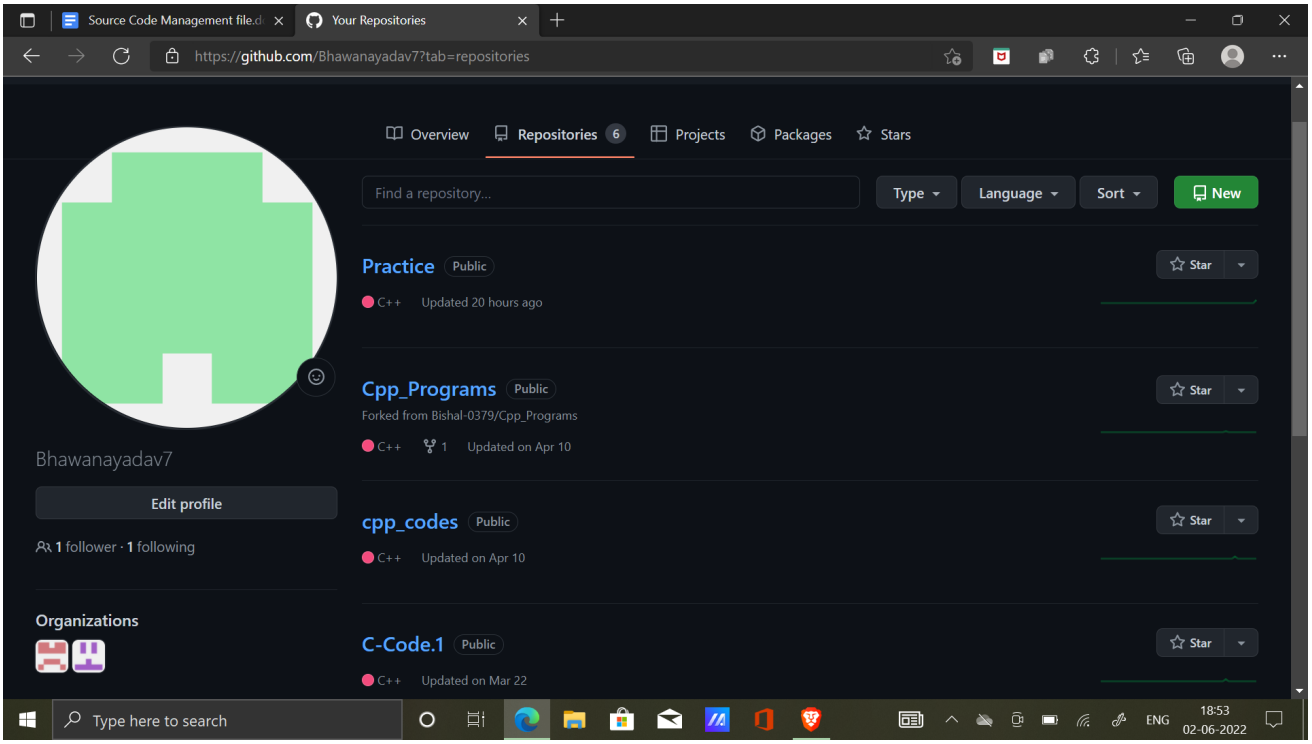
Project Work

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

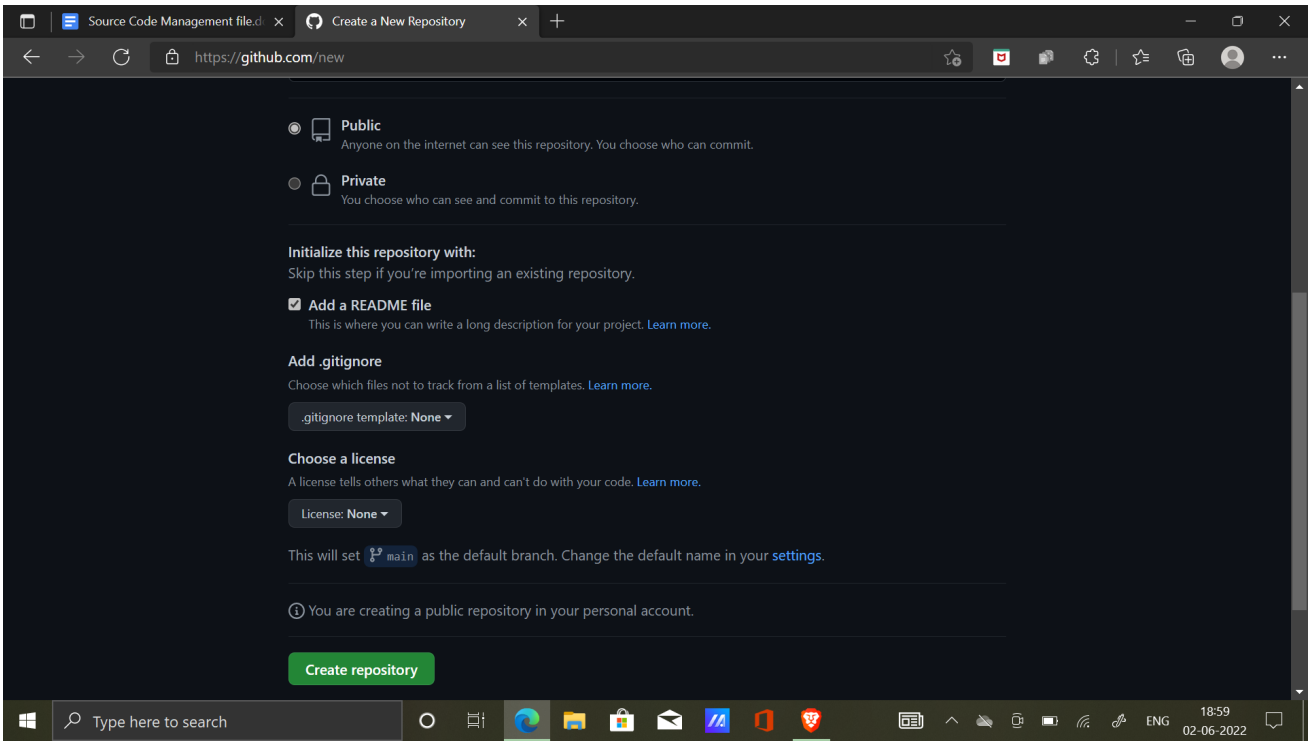
Aim: Create a distributed repository and add team members in the project team.

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.

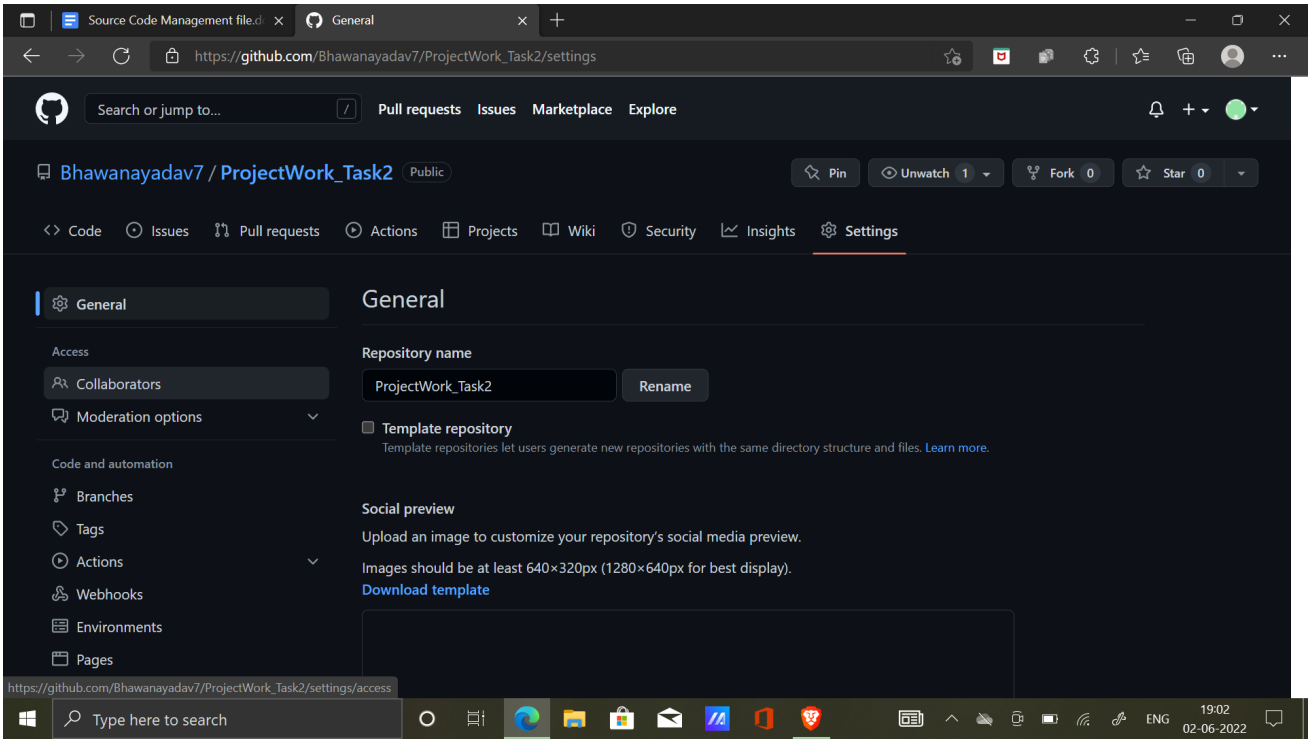


Step 2:

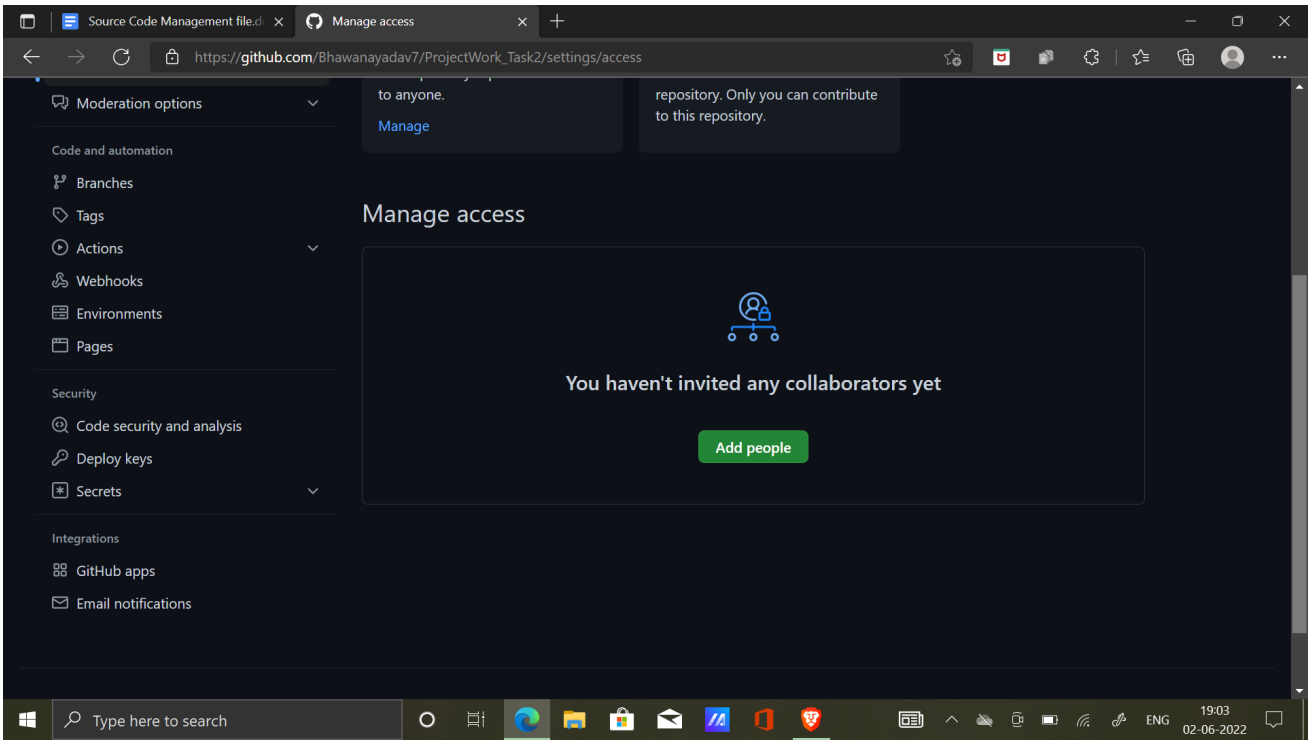
•Enter the Repository name and add description if you want.
• Select if you want the repository to be public or private.
• Click on “Create Repository”.



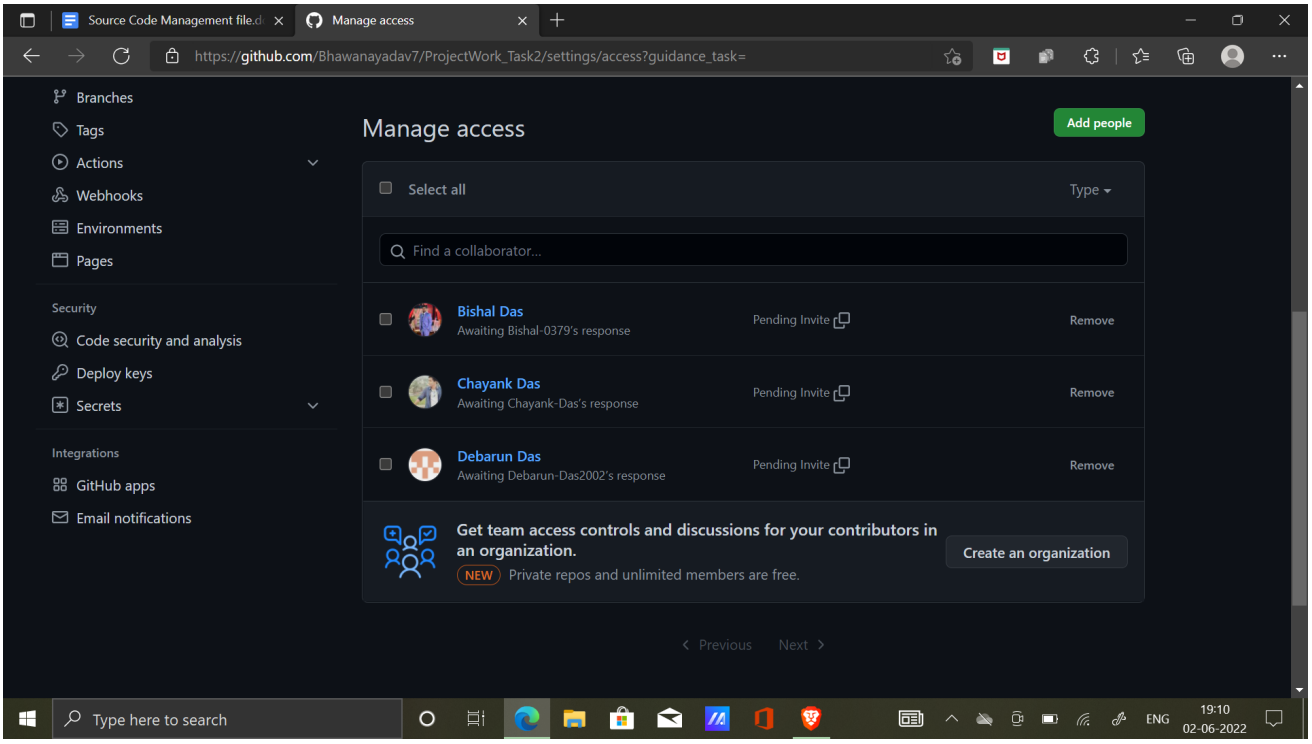
Step 3: click on settings and after that go to the collaborators option to add team members



Step 4: Add members by clicking on add people.

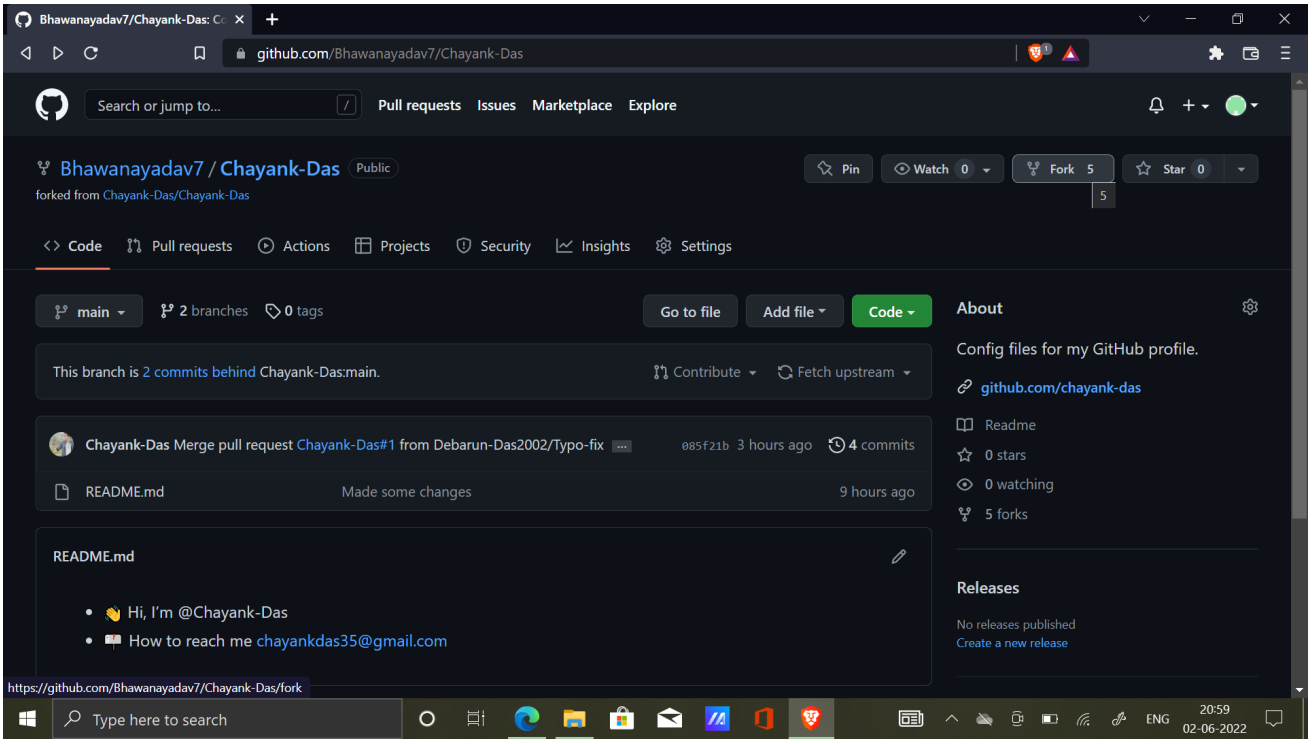


Step 5: under Manage access you can see who has or hasn't accepted your request.

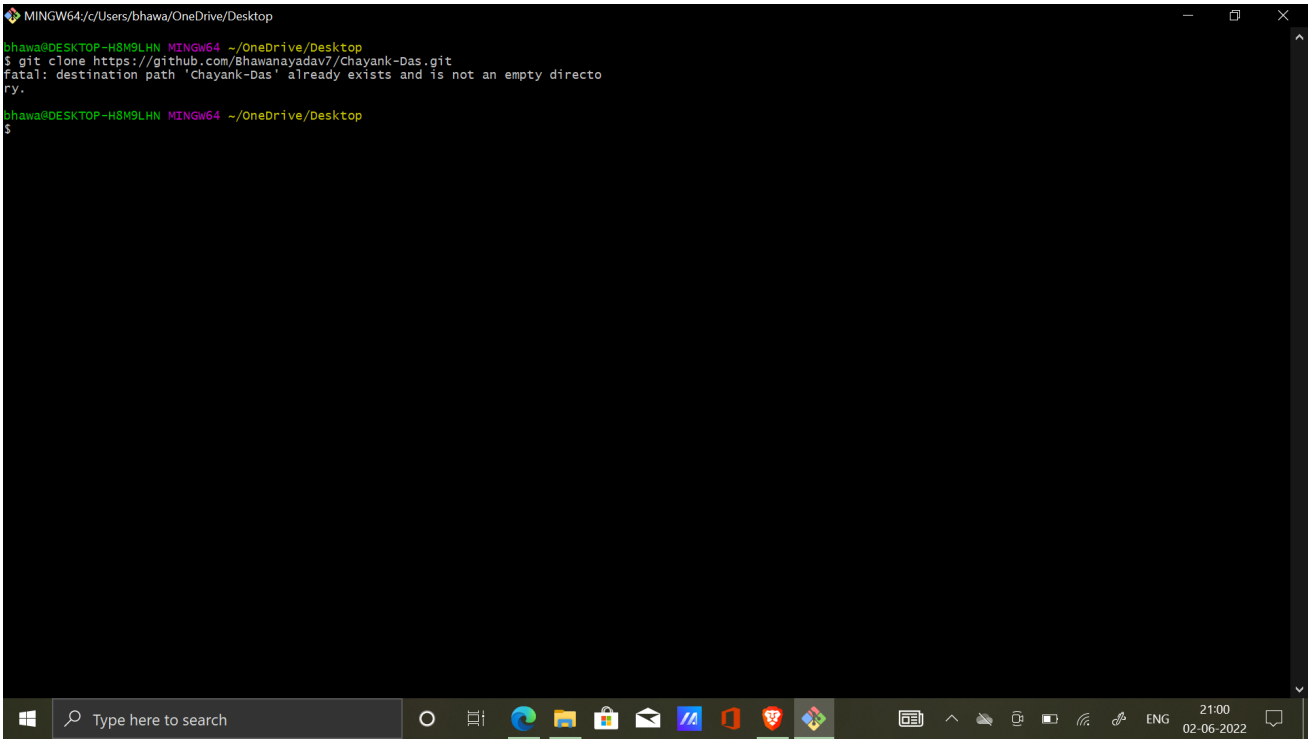


Aim: Open and Close pull request on team members repository.

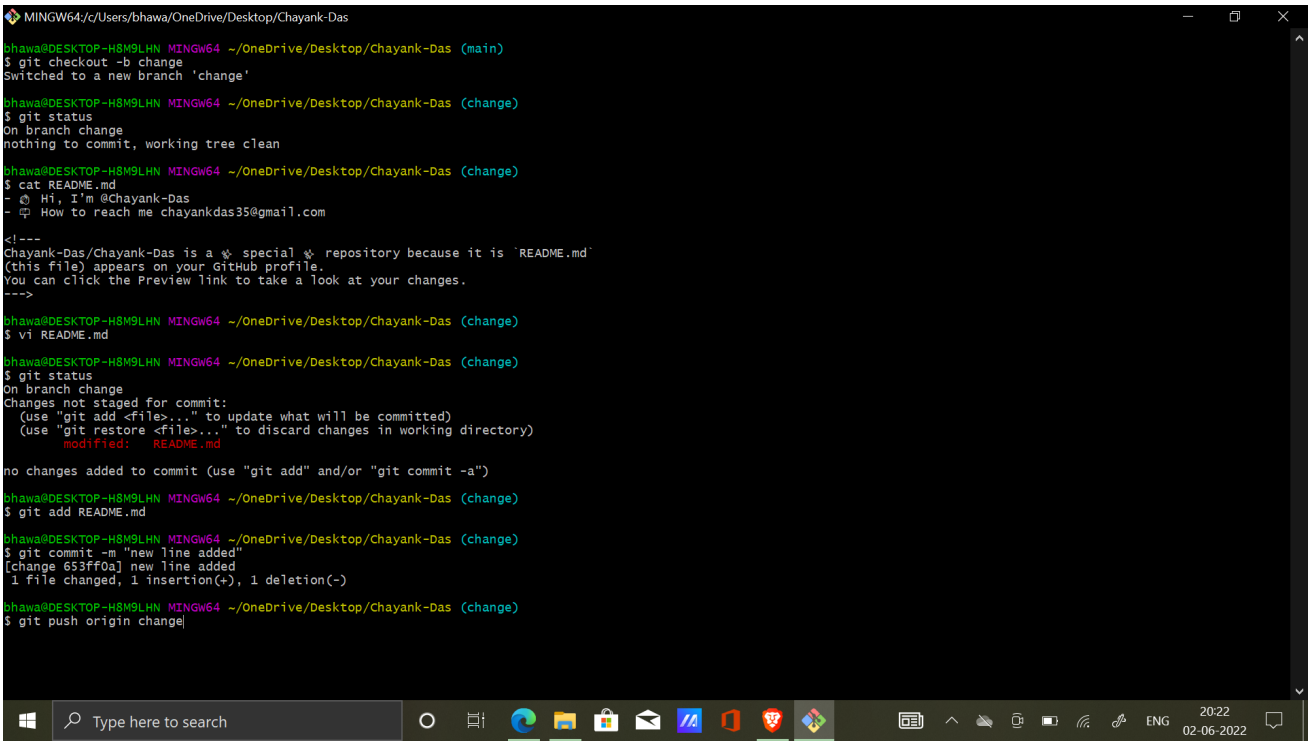
Step 1: Fork the repository you want to amend changes to.



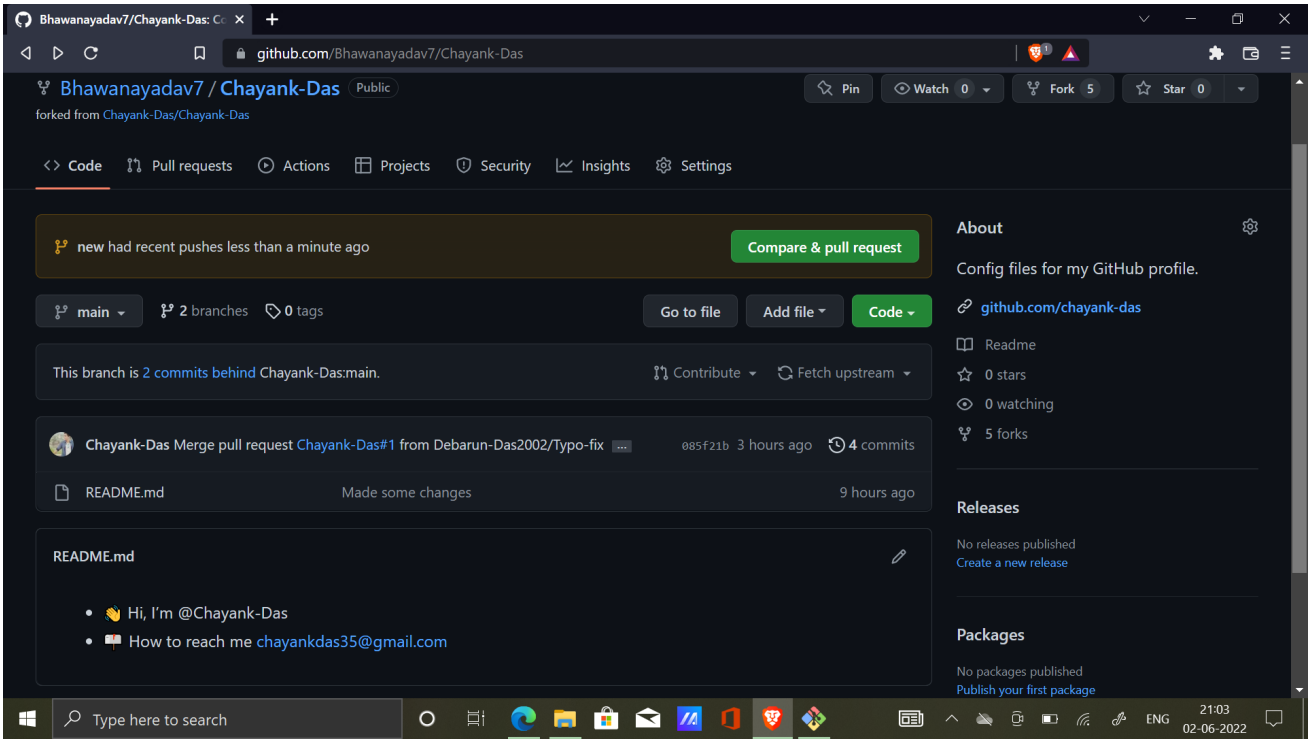
Step 2: Clone to your local computer.



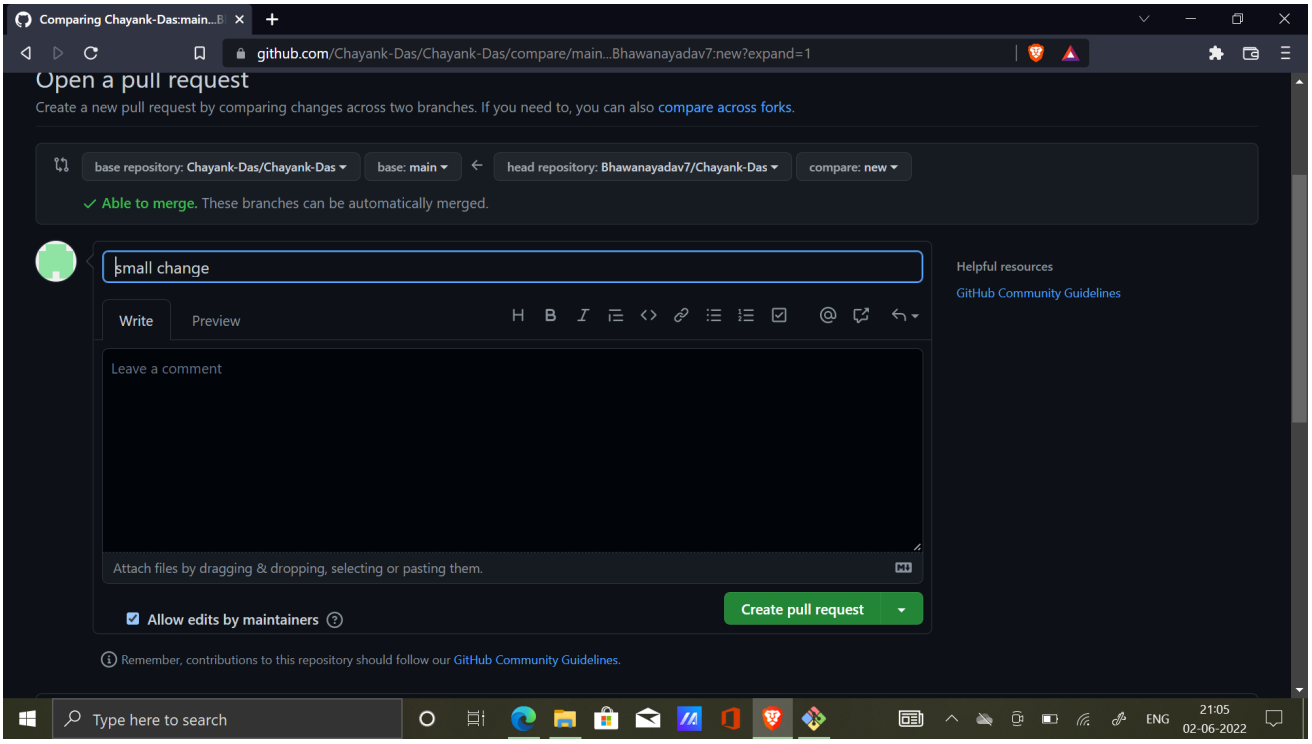
Step 3: make some changes in any of the files or add new file and push it.

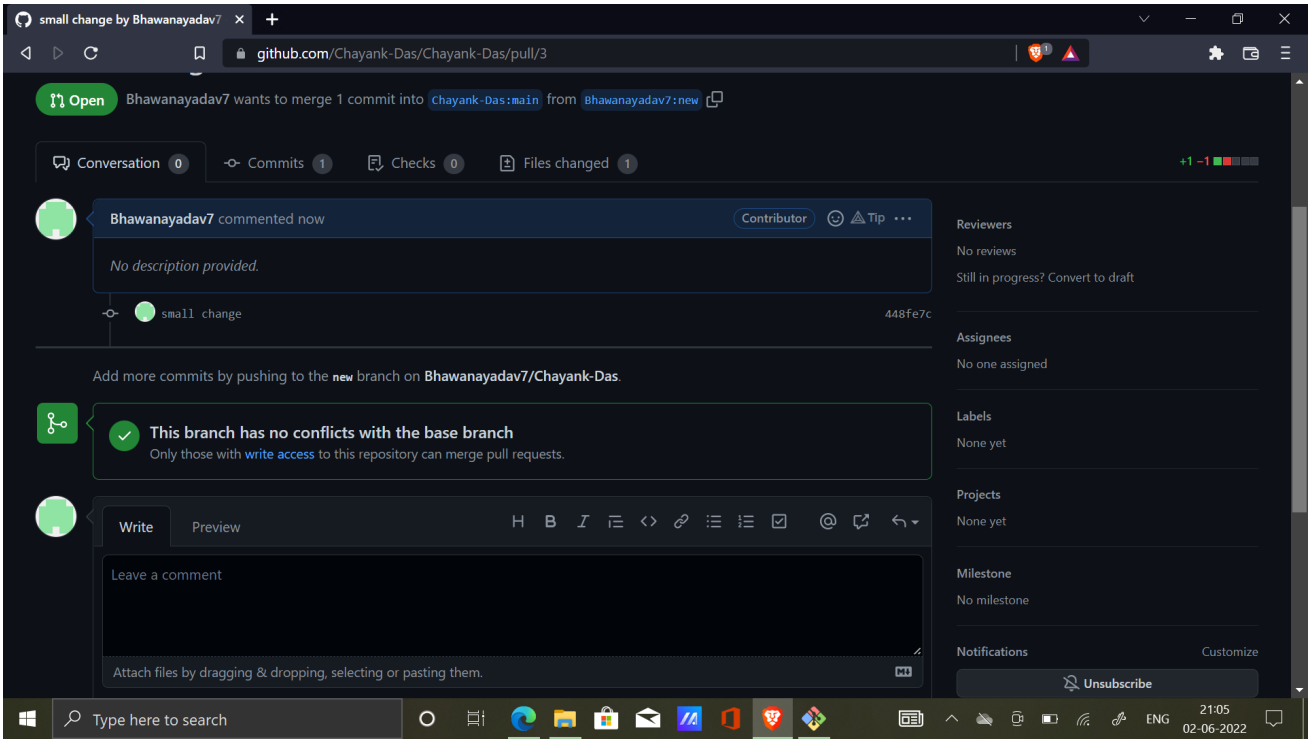


Step 4: compare and pull the merge request. As we are team members no open request will be needed.



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


Comparing Chayank-Das:main...B
github.com/Chayank-Das/Chayank-Das/compare/main...Bhawanayadav7:new?expand=1
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
Remember, contributions to this repository should follow our GitHub Community Guidelines.
Helpful resources
GitHub Community Guidelines
Type here to search
ENG
21:05
02-06-2022


small change by Bhawanayadav7
github.com/Chayank-Das/Chayank-Das/pull/3
Open
Bhawanayadav7 wants to merge 1 commit into Chayank-Das:main from Bhawanayadav7:new
Conversation 0
Commits 1
Checks 0
Files changed 1
+1 −1
Bhawanayadav7 commented now
Contributor
Tip
No description provided.
small change
448fe7c
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
ENG
21:05
02-06-2022

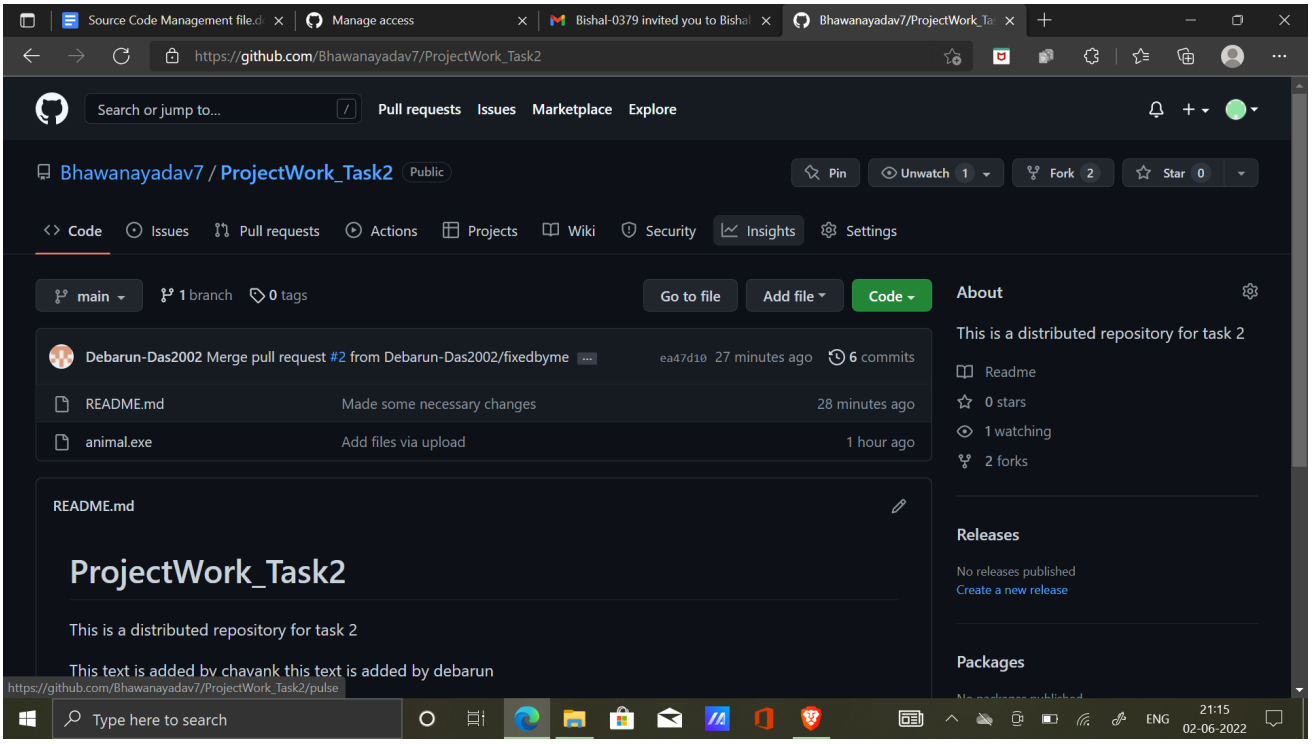
Aim: Publish and print Network graph

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.

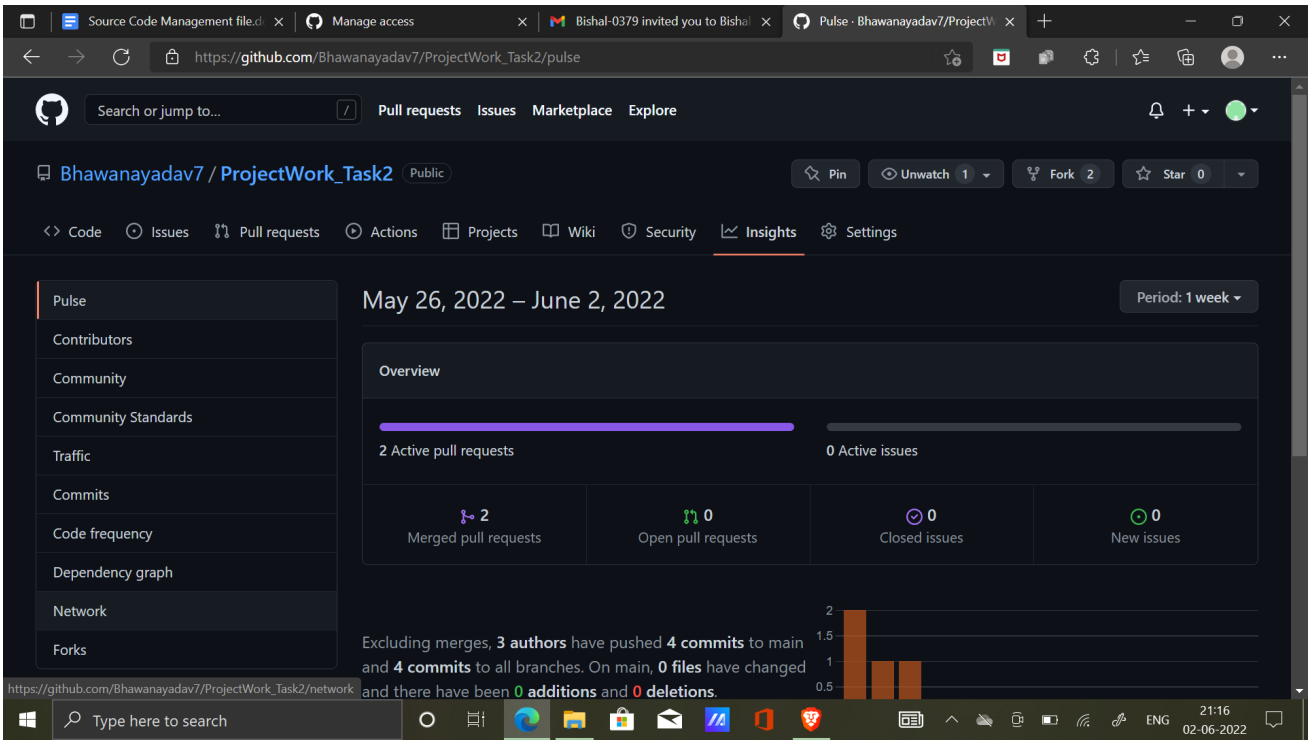
A repository's graphs give you information on traffic, projects that depend on the repository, contributors and commits to the repository, and a repository's forks and network. If you maintain a repository, you can use this data to get a better understanding of who's using your repository and why they're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

Steps to access network graphs of respective repository

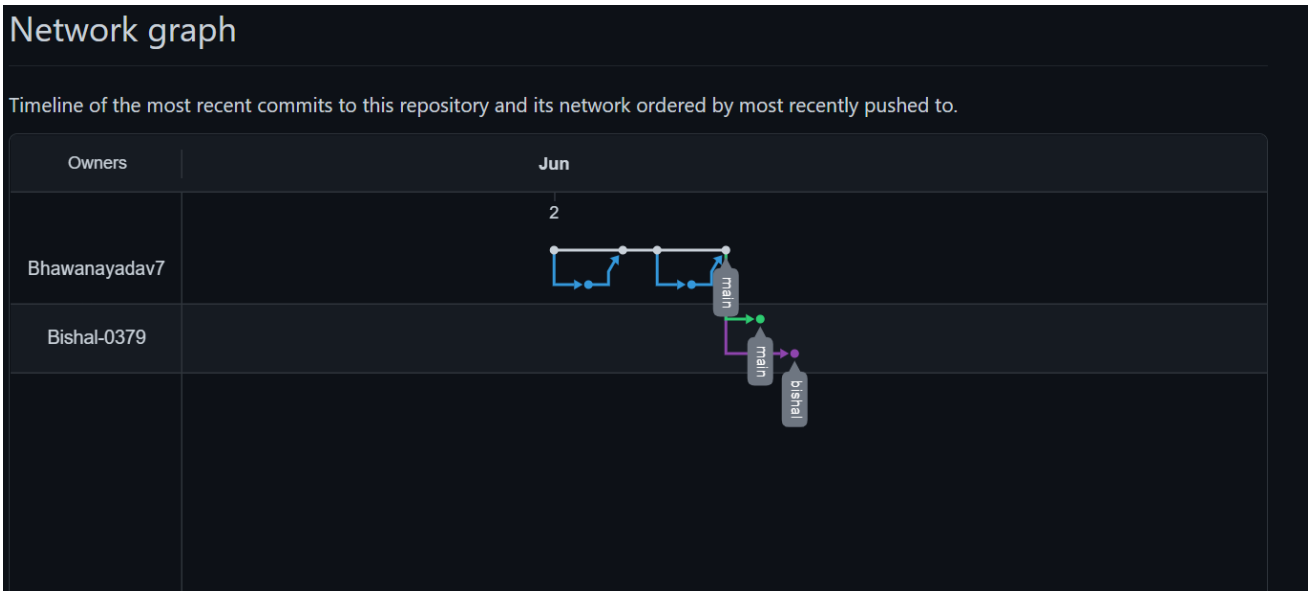
1. On GitHub.com navigate to the main page of the repository.
2. Under your repository name, click Insights.



3.At the left sidebar, click on 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.