

PRACTICAL-1

AIM: Installing and Setting up Git client

Theory:

What is Git?

Git is a free and open-source version control system used to handle small to very large projects efficiently. This is also used for tracking changes in any set of files and usually helps in coordinating work among members of a team. Hence, enables multiple developers to work together on non-linear development.

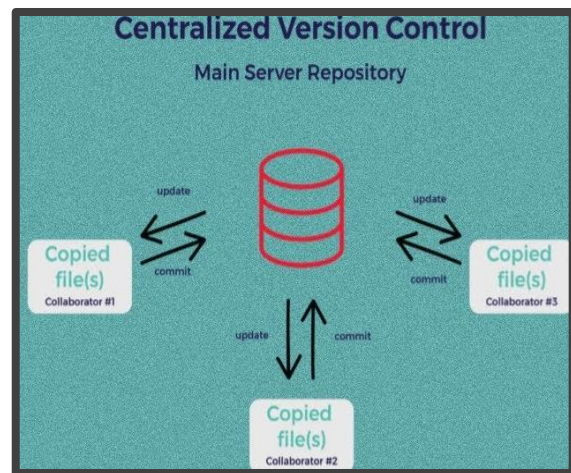
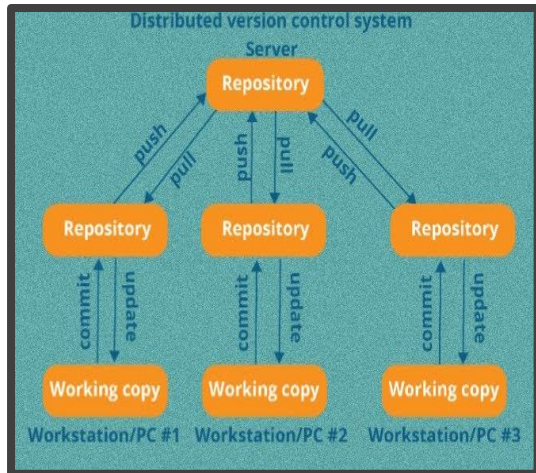
History of VCS:

The very first Version Control System was created in 1972 at Bell Labs where they also developed UNIX. The first one was called SCCS (Source Code Control System).

Some types of Version Control Systems are:

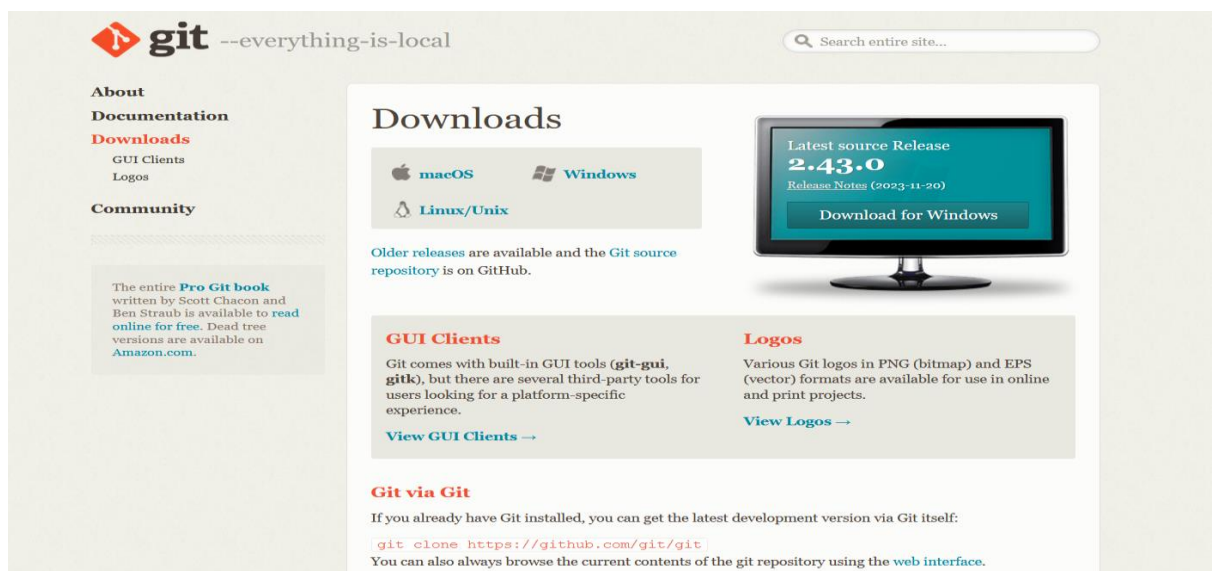
- **Local VCS:** No internet is needed because it uses a database to keep and track of files.
- **Centralized VCS:** Centralized version control systems are based on the idea that there is a single “central” copy of your project somewhere (probably on a server), and programmers will “commit” their changes to this central copy. This simply means recording the change in the central system (OS).

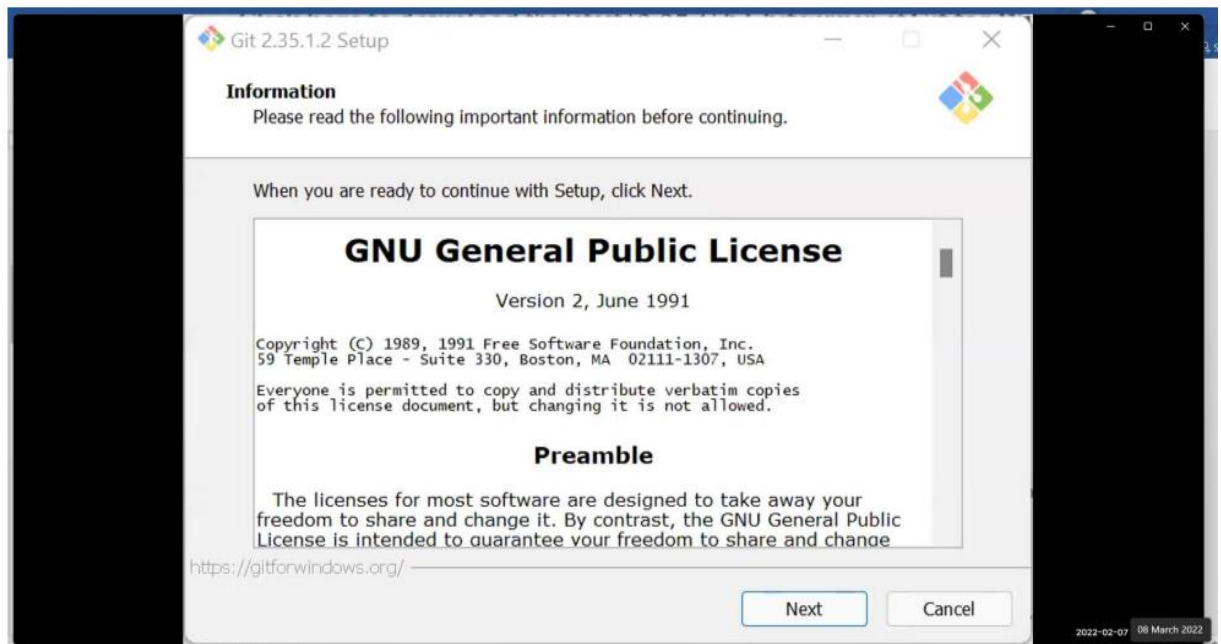
- **Distributed VCS:** A type of version control where the complete codebase including its full version history is mirrored on every developer's computer.



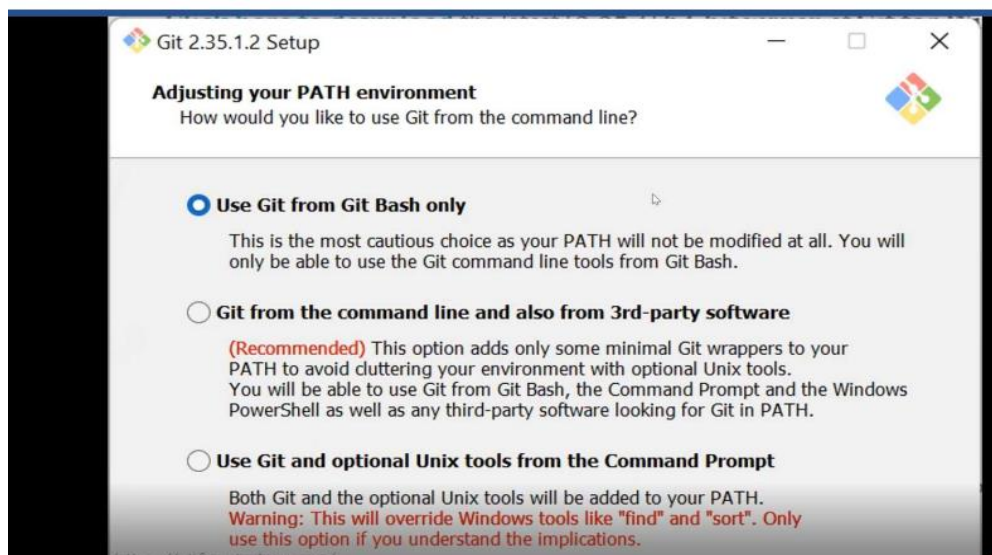
Steps to install Git on Windows:

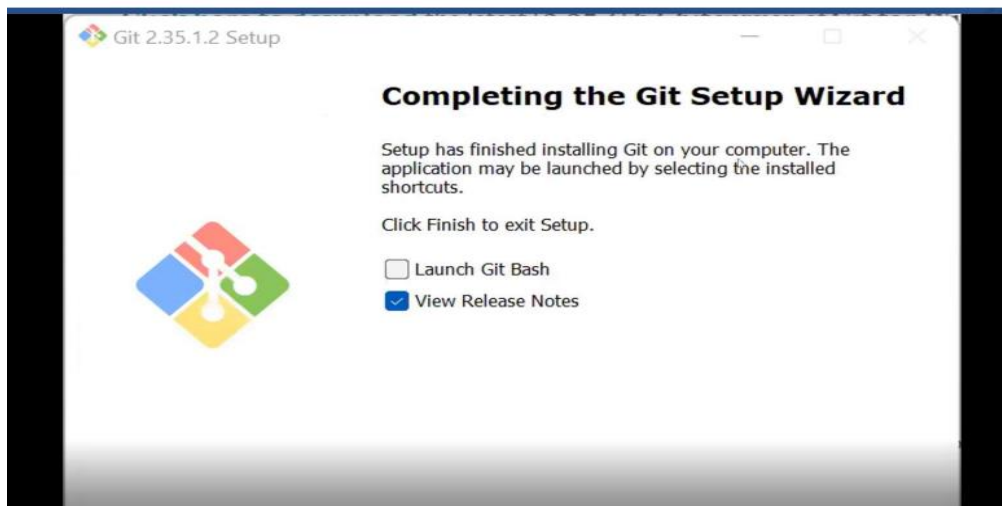
1. Visit the page <https://git-scm.com/downloads>
2. Then click on Installation Git and click on whatever system you want, available are three- Windows, Apple and Linux.





3. After some more simple and easy settings and choosing your favourable environment and doing some SSH settings, it finally starts exporting the files in system and completes the Git hub wizard.





4. Git bash is installed in system.
5. You can also check the version of installed software by checking git version.

A screenshot of a terminal window with a dark background. The title bar shows 'MINGW64:/c/Users/kriti'. The prompt is 'kriti@Kriti_Vivobook MINGW64 ~ (master)'. The user has entered the command '\$ git --version' and the output is 'git version 2.43.0.windows.1'. The prompt is now '\$ |' with a cursor.

PRACTICAL-2

AIM: Setting up Git account

Theory:

What is GitHub?

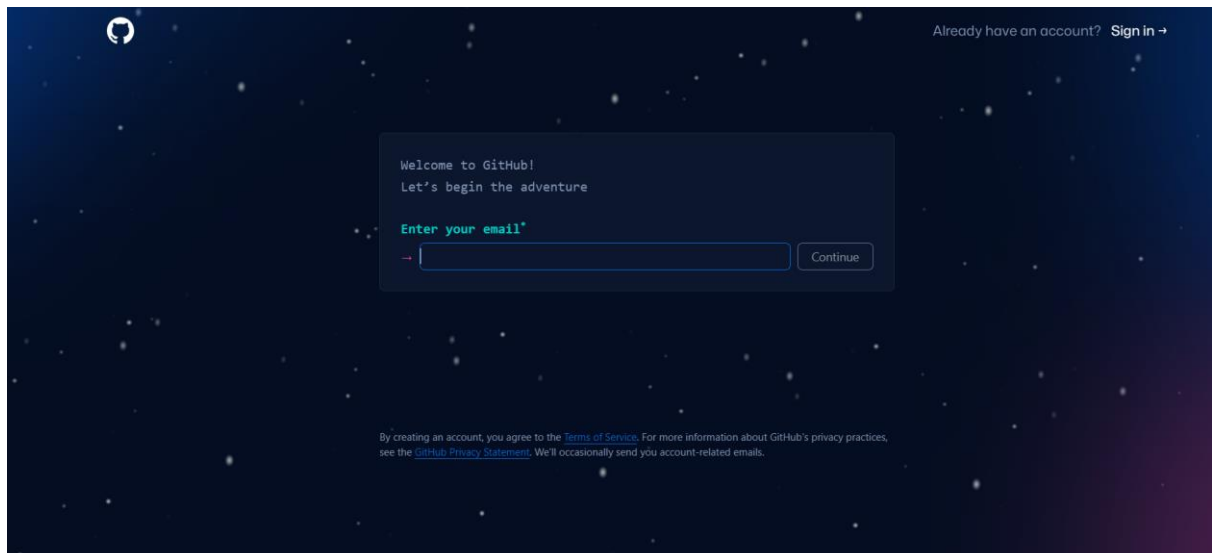
GitHub is a code hosting platform for version control and collaboration. GitHub is a development platform inspired by the way you work. From open source to business, we can host and review code, manage projects, and build software alongside 36 million developers.

Advantages:

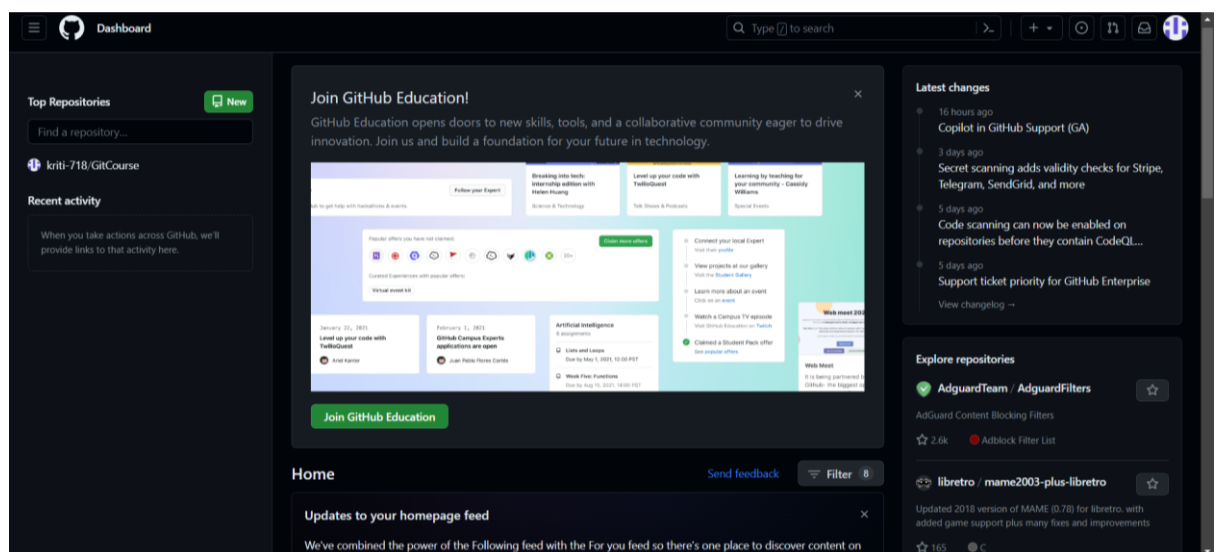
- Documentation.
- Showcase your work.
- Markdown.
- GitHub is a repository.
- Track changes in your code across versions.
- Integration options.

Steps to create an account on GitHub:

1. To sign up for an account on GitHub.com, navigate to <https://github.com/>.
2. Using your e-mail sign in to GitHub account and create a strong password to keep your account strong.



3. An interface like following would appear.



For linking GitHub with Git Bash:

- Username-

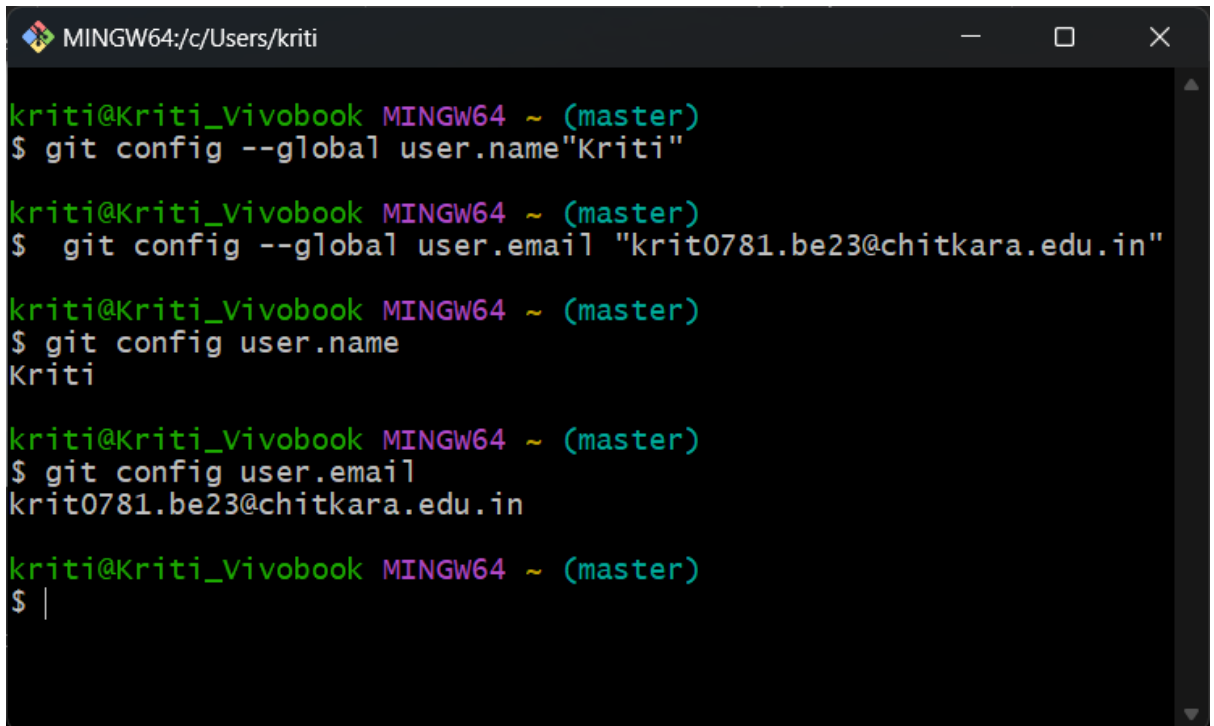
`git config --global user.name "username in GitHub"`

- Email-

git config --global user.email "your email in GitHub"

Check Username & Email:

- git config user.name
- git config user.email

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/kriti'. The window shows a series of git configuration commands and their outputs. The prompt is 'kriti@Kriti_Vivobook MINGW64 ~ (master)'. The commands and outputs are: 1. '\$ git config --global user.name "Kriti"' followed by 'kriti@Kriti_Vivobook MINGW64 ~ (master)'. 2. '\$ git config --global user.email "krit0781.be23@chitkara.edu.in"' followed by 'kriti@Kriti_Vivobook MINGW64 ~ (master)'. 3. '\$ git config user.name' followed by 'Kriti'. 4. '\$ git config user.email' followed by 'krit0781.be23@chitkara.edu.in'. 5. '\$ |' followed by a cursor. The terminal has a dark background with green and purple text for the prompt and standard white text for commands and output.

```
MINGW64:/c/Users/kriti

kriti@Kriti_Vivobook MINGW64 ~ (master)
$ git config --global user.name "Kriti"

kriti@Kriti_Vivobook MINGW64 ~ (master)
$ git config --global user.email "krit0781.be23@chitkara.edu.in"

kriti@Kriti_Vivobook MINGW64 ~ (master)
$ git config user.name
Kriti

kriti@Kriti_Vivobook MINGW64 ~ (master)
$ git config user.email
krit0781.be23@chitkara.edu.in

kriti@Kriti_Vivobook MINGW64 ~ (master)
$ |
```

PRACTICAL-3

AIM: Generate logs on GitHub

Theory:

Git Logs: The git log command shows a list of all the commits made to a repository. You can see the hash of each Git commit, the message associated with each commit, and more metadata. This command is basically used for displaying the history of a repository.

Why do we need logs?

Git log is a utility tool to review and read a history of everything that happens to a repository. Anything we change at what time, by which log, everything is getting recorded in git logs.

Steps:

1. Using “cd” command navigate through any folder in your PC. Check the files in that folder through ‘ls’ command.


```
MINGW64/c/users/kriti
kriti@Kriti_Vivobook MINGW64 ~ (master)
$ cd C:/
kriti@Kriti_Vivobook MINGW64 /c
$ cd users
kriti@Kriti_Vivobook MINGW64 /c/users
$ cd kriti
kriti@Kriti_Vivobook MINGW64 /c/users/kriti (master)
$ ls
AppData/          Links/
'Application Data'@ 'Local Settings'@
Contacts/         Music/
Cookies@          'My Documents'@
Documents/        NTUSER.DAT
Downloads/        NTUSER.DAT{51efd922-2c25-11ee-bdc7-cffc9bfe8e7}.TM.blf
Favorites/        NTUSER.DAT{51efd922-2c25-11ee-bdc7-cffc9bfe8e7}.TMContainer000000000000000001.regtrans-ms
Game1.txt         NTUSER.DAT{51efd922-2c25-11ee-bdc7-cffc9bfe8e7}.TMContainer000000000000000002.regtrans-ms
Game2.txt         NetHood@
GitCourse/        OneDrive/
KritiGit/         Pictures/
PrintHood@        f1.txt
Recent@           file.txt/
'Saved Games'/    ntuser.dat.LOG1
Searches/         ntuser.dat.LOG2
SendTo@          ntuser.ini
'Start Menu'@    scm.txt
Templates@       untitled
Untitled.ipynb   untitled.txt
Videos/          xyz.txt
anaconda3/
credentials.txt
```

2. Using “git add” command add any one file from the folder.
3. Commit the file using “git commit -m ‘type any message here’”.
4. Check the status using “git status”.

```
MINGW64/c/users/kriti
kriti@Kriti_Vivobook MINGW64 /c/users/kriti (master)
$ git add scm.txt
kriti@Kriti_Vivobook MINGW64 /c/users/kriti (master)
$ git commit -m "This is the new file to commit."
[master 60c6704] This is the new file to commit.
 2 files changed, 8 insertions(+)
 create mode 100644 f1.txt
 create mode 100644 scm.txt
kriti@Kriti_Vivobook MINGW64 /c/users/kriti (master)
$ git status
warning: could not open directory 'Application Data/': Permission denied
warning: could not open directory 'Cookies/': Permission denied
warning: could not open directory 'Local Settings/': Permission denied
warning: could not open directory 'My Documents/': Permission denied
warning: could not open directory 'NetHood/': Permission denied
warning: could not open directory 'PrintHood/': Permission denied
warning: could not open directory 'Recent/': Permission denied
warning: could not open directory 'SendTo/': Permission denied
warning: could not open directory 'Start Menu/': Permission denied
warning: could not open directory 'Templates/': Permission denied
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .bash_history
        .condarc
        .continuum/
        .gitconfig
        .ipynb_checkpoints/
        .ipython/
        .jupyter/
        .lessht
        .ssh/
```

5. Use “git log” to generate logs.

```
kriti@Kriti_vivobook MINGW64 /c/users/kriti (master)
$ git log
commit 60c67048e51a77d2e79ac7fa1d155cde6ea1f598 (HEAD -> master)
Author: Kriti <krit0781.be23@chitkara.edu.in>
Date: Sat Feb 10 15:30:50 2024 +0530

    This is the new file to commit.
```

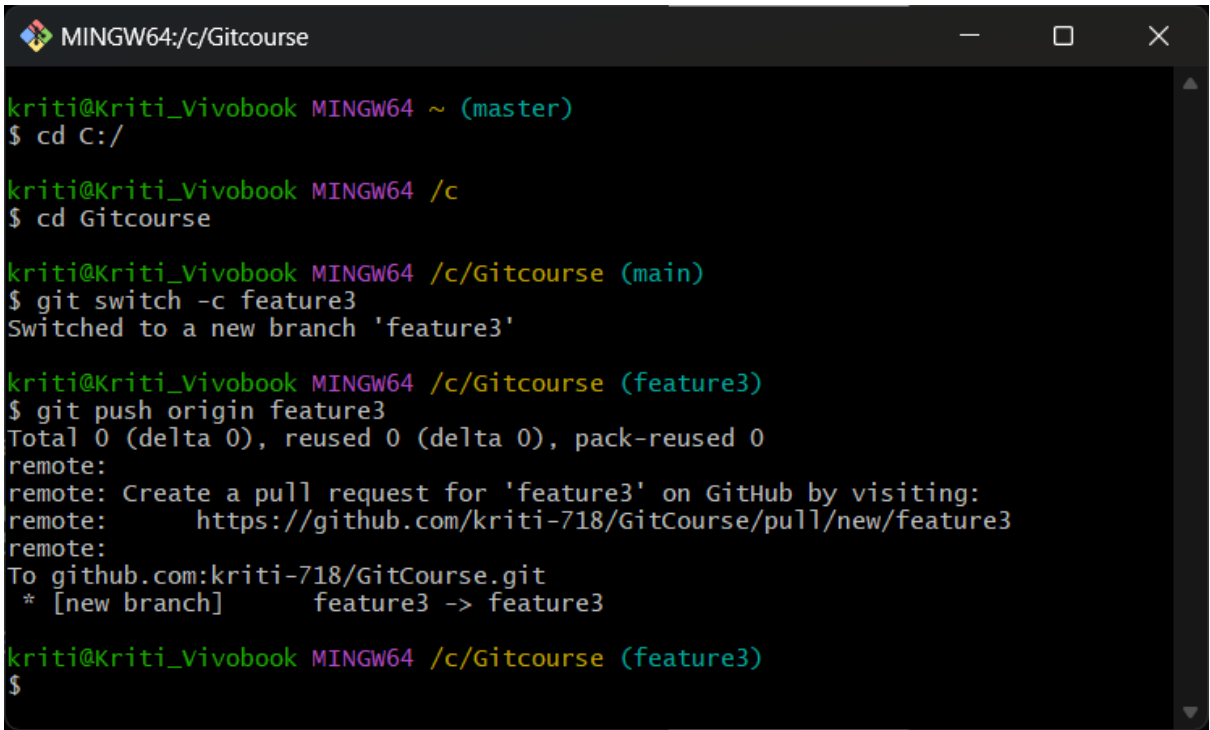
PRACTICAL-4

AIM: Creating and Visualizing the Branches On Git Client

How to create branches?

The main branch in which we are working is master branch. you can use the “git branch” command with the branch name and the commit SHA for the new branch.

1. To create a new branch use “git switch -c ‘name of branch’”.
2. Use command “git push origin ‘name of branch’” to push it to your git repository on GitHub.



```
MINGW64:/c/Gitcourse
kriti@Kriti_Vivobook MINGW64 ~ (master)
$ cd C:/


kriti@Kriti_Vivobook MINGW64 /c
$ cd Gitcourse

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (main)
$ git switch -c feature3
Switched to a new branch 'feature3'

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (feature3)
$ git push origin feature3
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature3' on GitHub by visiting:
remote:   https://github.com/kriti-718/GitCourse/pull/new/feature3
remote:
To github.com:kriti-718/GitCourse.git
 * [new branch]      feature3 -> feature3

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (feature3)
$
```

3. To check the number of branches, use “git branch”.

A terminal window titled 'MINGW64:/c/Gitcourse' showing the command 'git branch' being executed. The output lists several branches: 'feature1', '* feature3' (the current branch), 'flag', 'main', and 'mybranch'.

```
kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (feature3)
$ git branch
feature1
* feature3
flag
main
mybranch

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (feature3)
$ |
```

4. To change the present working branch use “git switch ‘name of branch’”.

A terminal window titled 'MINGW64:/c/Gitcourse' showing the command 'git switch flag' being executed. The output indicates a successful switch to the 'flag' branch and lists modified files: 'G9_6feb', 'README.md', and 'README1.md'.

```
kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (feature3)
$ git switch flag
Switched to branch 'flag'
M      G9_6feb
M      README.md
M      README1.md

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ |
```

Visualizing branches:

For visualizing, we have to create a new file in the branch that we made “new” instead of the master branch. After this we have to do three step architecture that is working directory, staging area and git repository.

Firstly, change the branch from master to “new”. Using “echo” command create a new file in the branch. Then add the file, check the status of the file and then commit it.

At last, check activities with the help of “git log” command.

```
MINGW64:/c/Gitcourse

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ echo "This is fourth practical in scm file">>practical4

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ git add practical4

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ git status
On branch flag
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   practical4

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:   G9_6feb
    modified:   README.md
    modified:   README1.md

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

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ git commit -m "Practical 4"
[flag d93977b] Practical 4
 1 file changed, 1 insertion(+)
 create mode 100644 practical4

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ git log
commit d93977bd634540630f3c59d94591c39631b1cb21 (HEAD -> flag)
Author: Kriti <krit0781.be23@chitkara.edu.in>
Date:   Sat Feb 10 15:39:54 2024 +0530

    Practical 4
```

PRACTICAL-5

AIM: Git lifecycle description

Theory:

Stages in GIT Life Cycle: Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area
- Git directory

Working Directory:

When a project is residing in our local system, we don't know whether the project is tracked by Git or not. In any of the case, this project directory is called our Working directory.

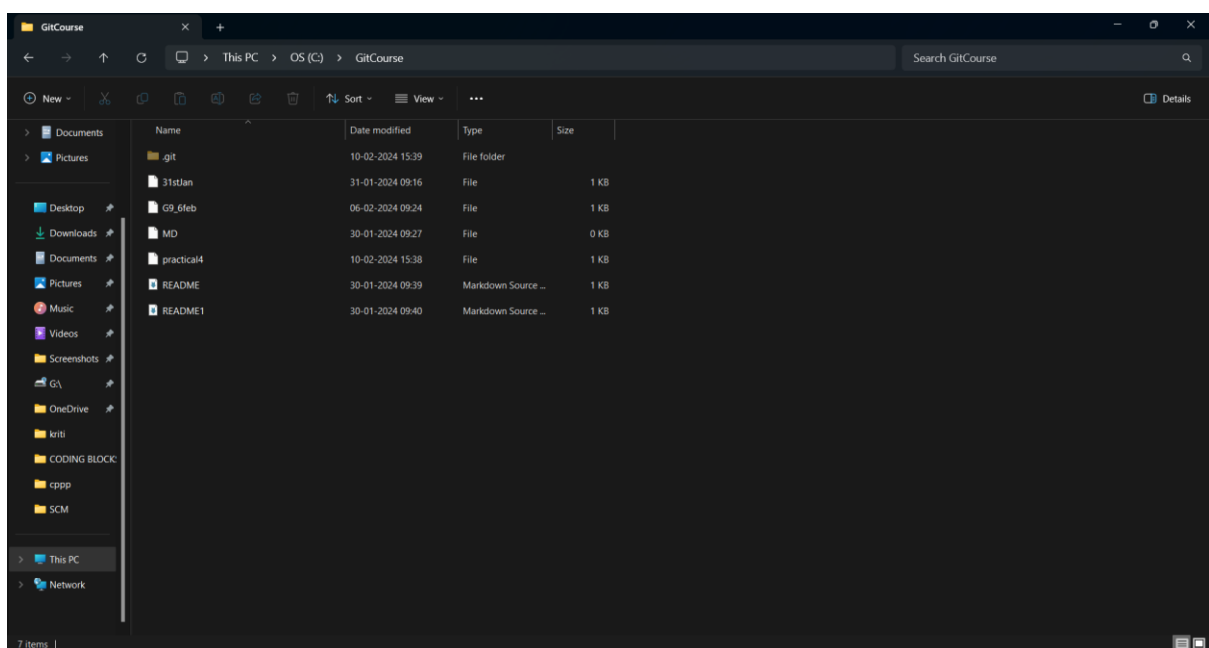
Staging Area:

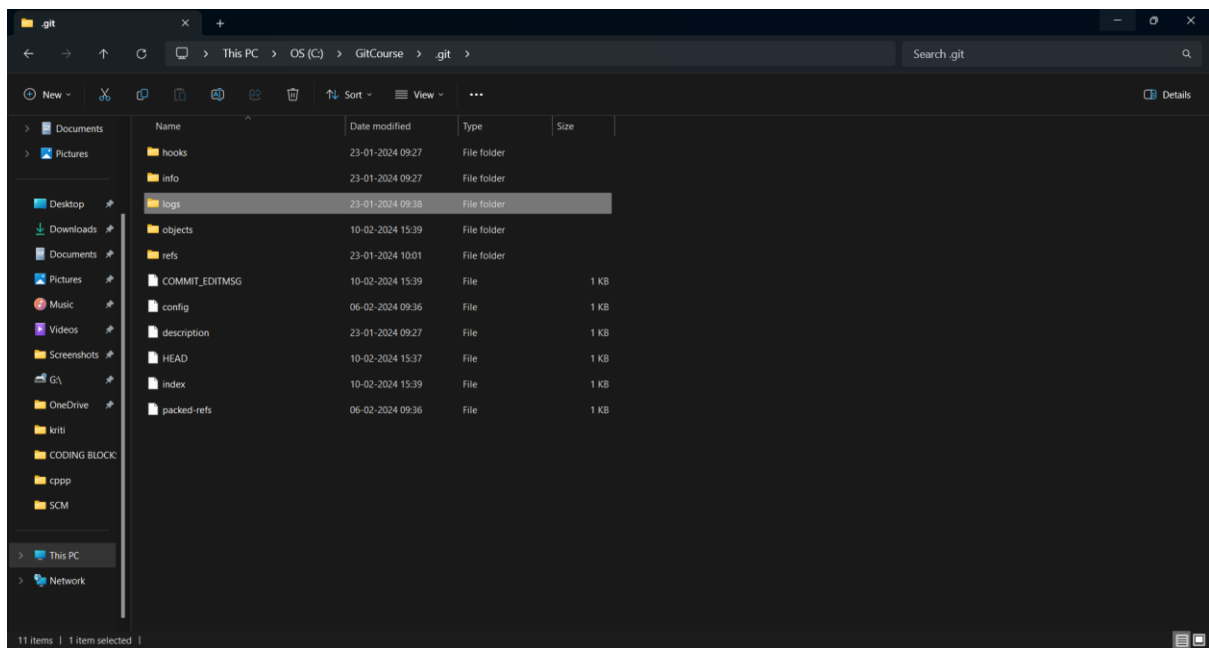
The staging area is like a rough draft space, it's where you can git add the version of a file or multiple files that you want to save in your next commit (in other words in the next version of your project)

Git Directory:

The `.git` folder contains all information that is necessary for the project and all information relating commits, remote repository address, etc. It also contains a log that stores the commit history. This log can help you to roll back to the desired version of the code

Remote Repository: Remote repositories are hosted on a server that is accessible for all team members - most likely on the internet or on a local network. Assessable and reachable by all.





```
MINGW64:/c/Gitcourse/practical5

kriti@Kriti_Vivobook MINGW64 ~ (master)
$ cd C:/

kriti@Kriti_Vivobook MINGW64 /c
$ cd Gitcourse

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ mkdir practical5

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse (flag)
$ cd practical5

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse/practical5 (flag)
$ git init
Initialized empty Git repository in C:/GitCourse/practical5/.git/

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse/practical5 (master)
$ echo "This is my SCM file practicals .">>practical5

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse/practical5 (master)
$ git add practical5

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse/practical5 (master)
$ git switch student
Switched to branch 'student'

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse/practical5 (student)
$ git branch
master
* student

kriti@Kriti_Vivobook MINGW64 /c/Gitcourse/practical5 (student)
$
```