Subject Name: **Source Code Management**

Subject Code: **24CSE0106**

Session: **2024-25**

Department: **CSE**

Submitted By:
Itishjot Singh
2410990366
G5

Submitted To:
Dr. Chetna Sharma

List of Programs

S. No	Program Title	Page No.
1	To install and configure Git Client on your local system	3-5
2	Setting up GitHub account ad adding collaborators on GitHub Repository	6-9
3	To merge two branches with a Git repository	10-13
4	To demonstrate push and pull operations in Git	14-17

Task 1.1

Practical 1
Aim: To install and configure Git Client on your local system.

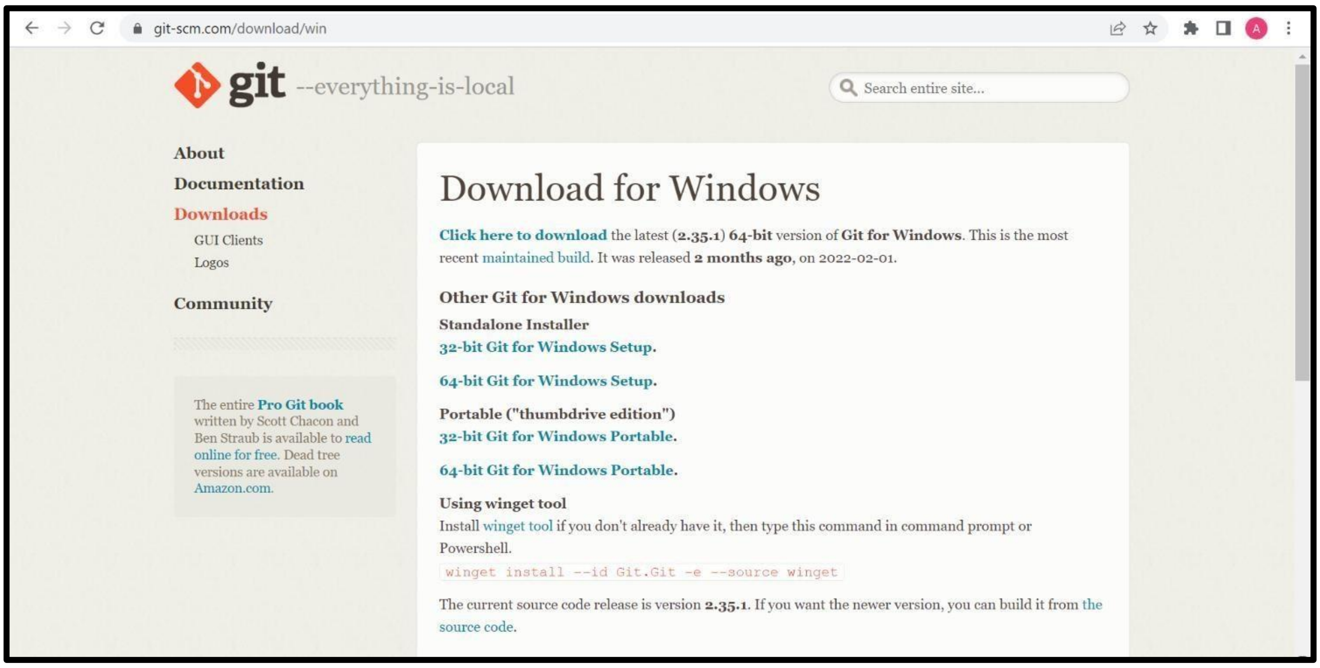
Theory:

Git is a distributed version control system used to track changes in source code. This practical focuses on setting up Git on your local system for effective version control.

Procedure:

○ Download Git from git-scm.com.
○ Install Git by following the setup wizard.
○ Open Git Bash and verify installation using the command: git --version.
○ Configure user details using the commands:
git config --global user.name "Your Name"
git config --global user.email "Your Email"

Snapshots of download:



git-scm.com/download/win

git --everything-is-local

Download for Windows

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

Other Git for Windows downloads

Standalone Installer
32-bit Git for Windows Setup.
64-bit Git for Windows Setup.

Portable ("thumbdrive edition")
32-bit Git for Windows Portable.
64-bit Git for Windows Portable.

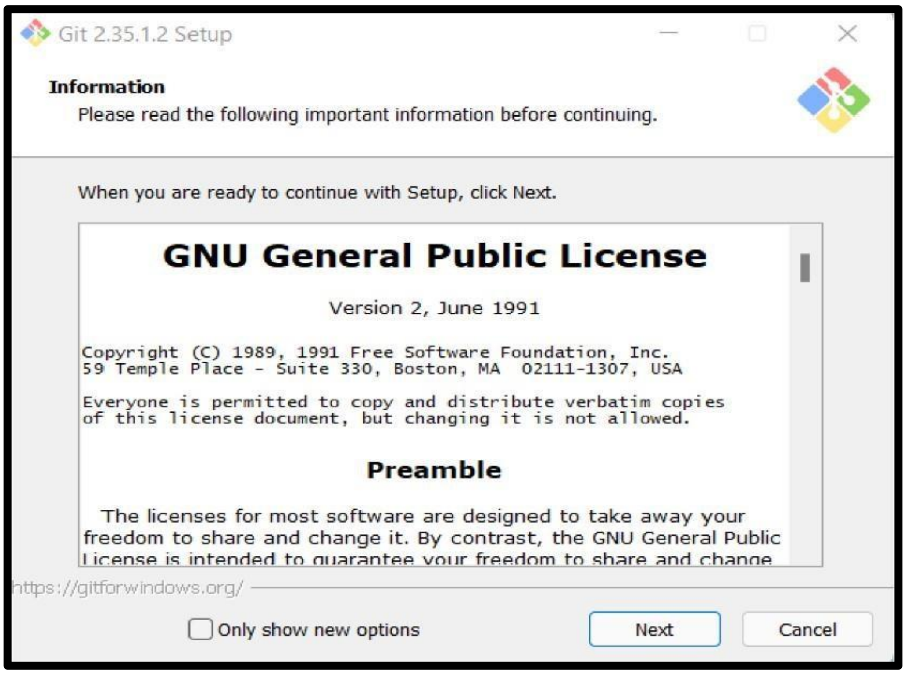
Using winget tool
Install winget tool if you don't already have it, then type this command in command prompt or Powershell.

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

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

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

Opted for "64-bit Git for Windows Setup"


Git 2.35.1.2 Setup

Information

Please read the following important information before continuing.

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

GNU General Public License

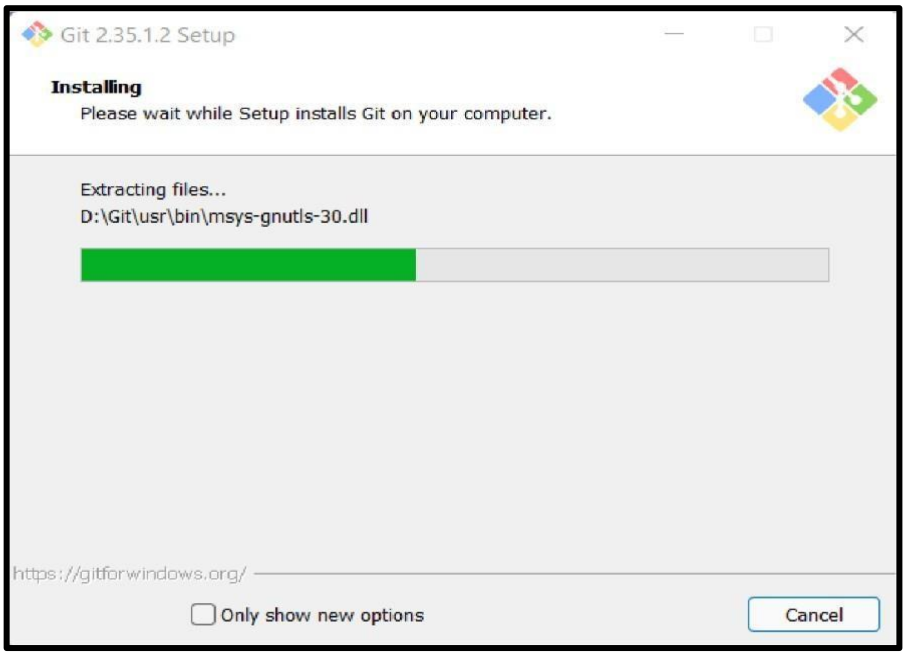
Version 2, June 1991

```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

Git Setup

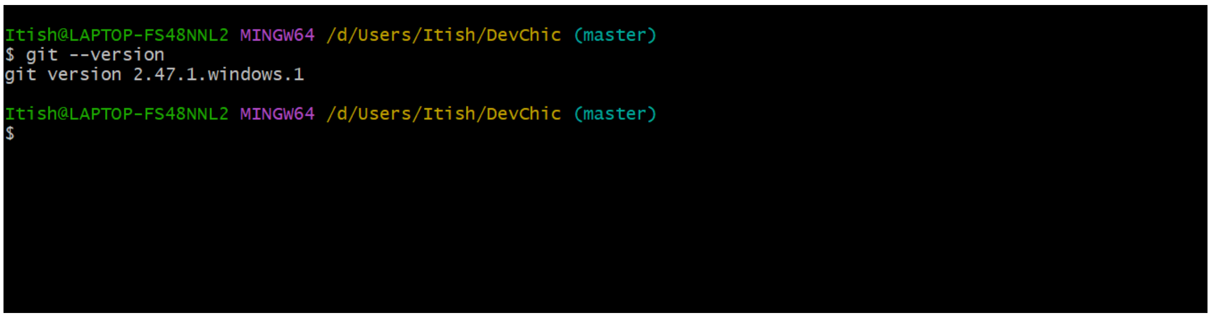

Git 2.35.1.2 Setup

Installing

Please wait while Setup installs Git on your computer.

Extracting files...
D:\Git\usr\bin\msys-gnutls-30.dll

Git Installation



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (master)
$ git --version
git version 2.47.1.windows.1

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (master)
$
```

Git Bash version

Practical 2

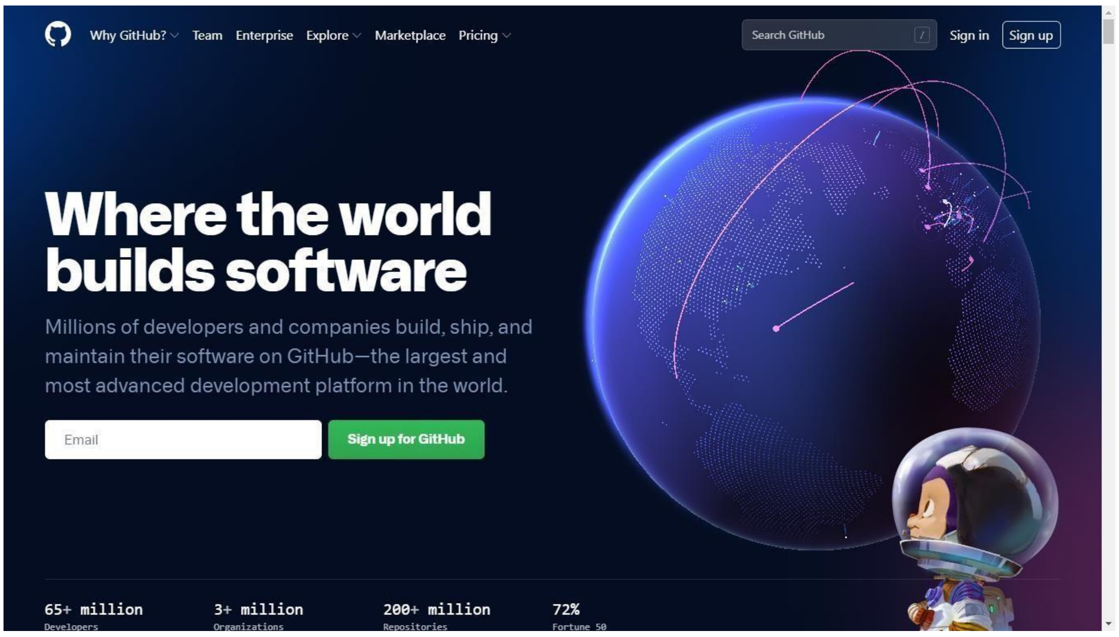
Aim: Setting up GitHub Account

Theory:
GitHub: GitHub is a website and cloud-based service (client) that helps an individual or developers to store and manage their code. We can also track as well as control changes to our or public code.

Advantages of GitHub: GitHub has a user-friendly interface and is easy to use. We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it will our team members, which can only be done by making a repository. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

Procedure:
To make an account on GitHub, we search for GitHub on our browser or visit https://github.com/signup. Then, we will enter our mail ID and create a username and password for a GitHub account.

Snapshots:



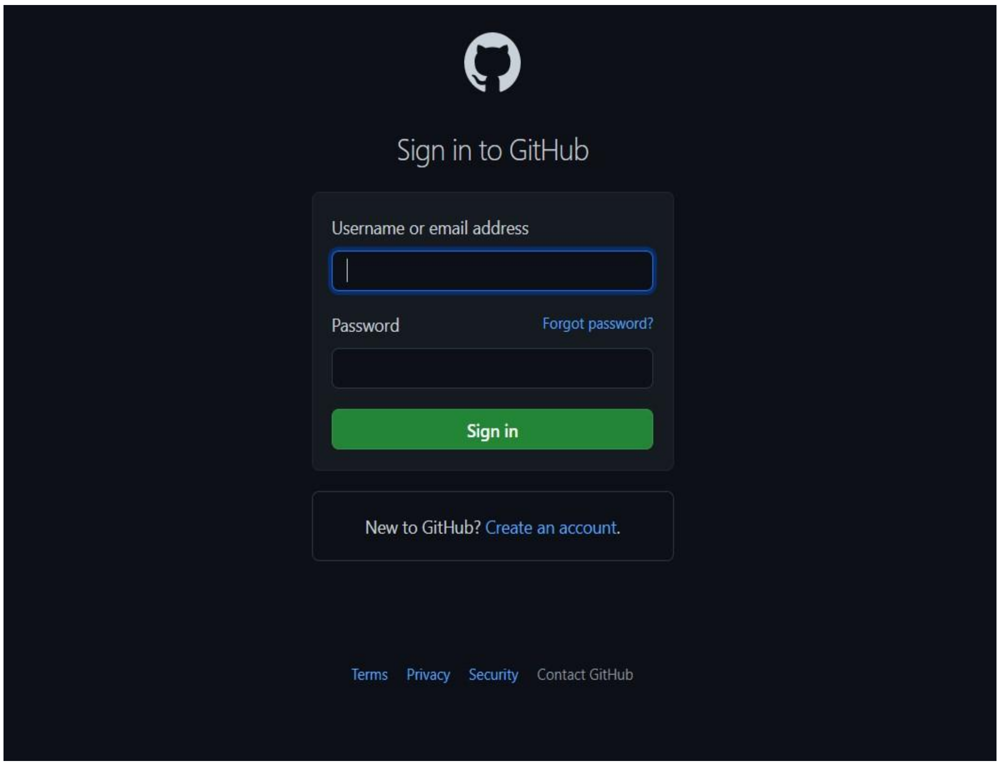
Where the world builds software

Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.

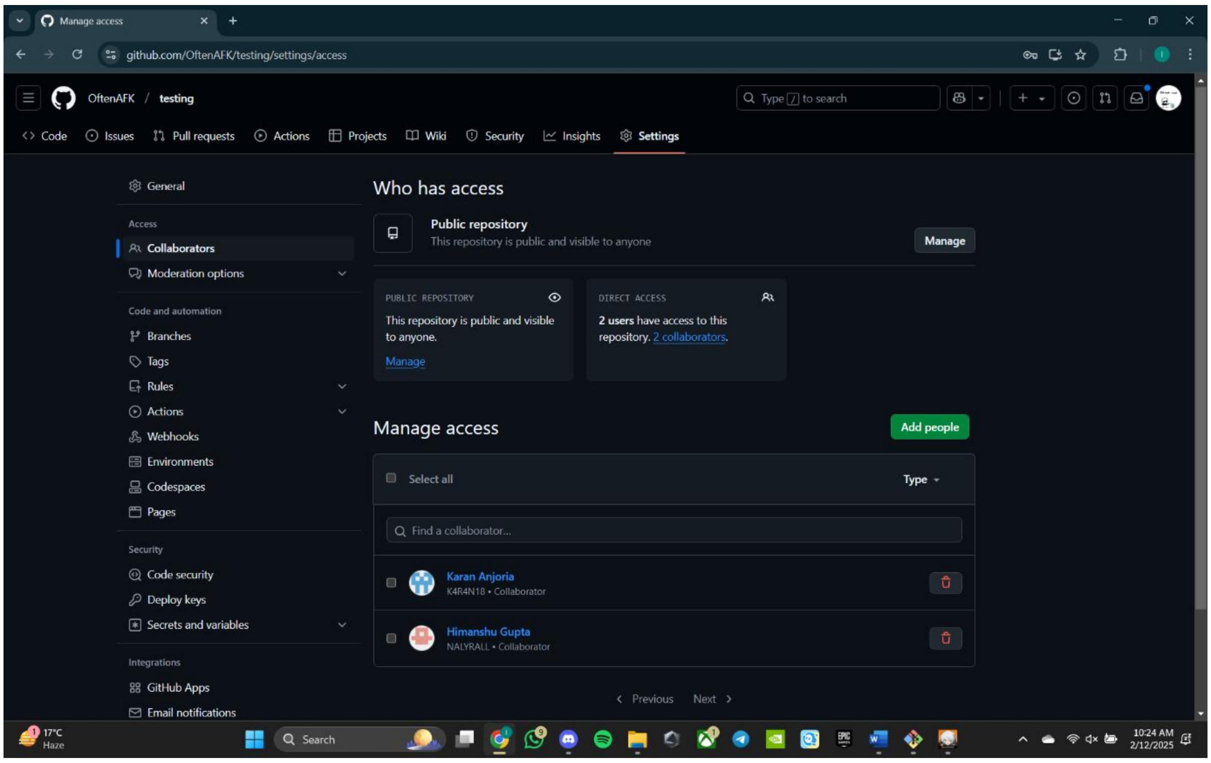
65+ million Developers | 3+ million Organizations | 200+ million Repositories | 72% Fortune 50

After visiting the link this type of interface will appear, if you already have an account, you can sign in and if not, you can create.

GitHub Login:



Adding Collaborators:



Who has access

Public repository
This repository is public and visible to anyone

PUBLIC REPOSITORY
This repository is public and visible to anyone.

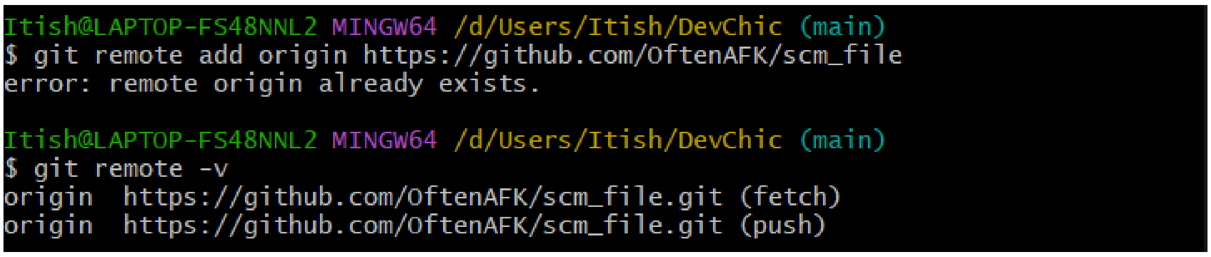
DIRECT ACCESS
2 users have access to this repository. 2 collaborators.

Manage access

Karan Anjoria
K4R4N18 • Collaborator

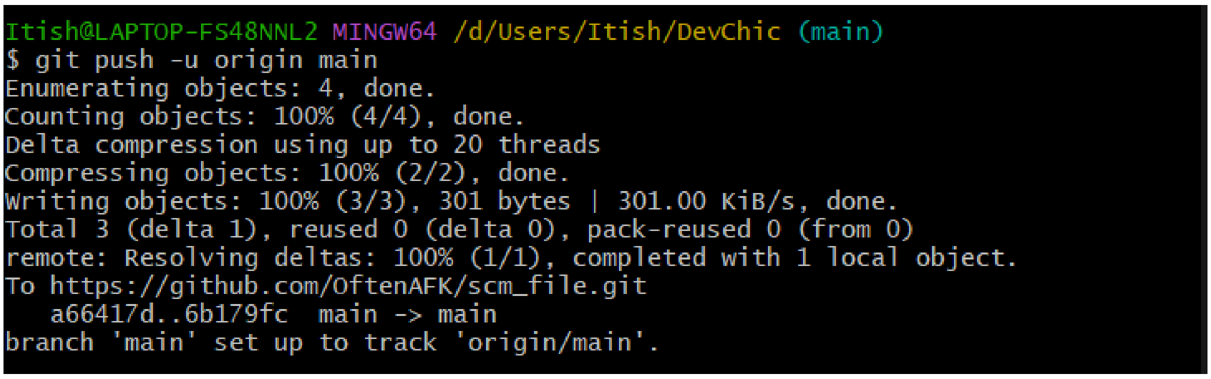
Himanshu Gupta
NALYRALL • Collaborator

List of Remote Repositories Associated with Local Git Repository:



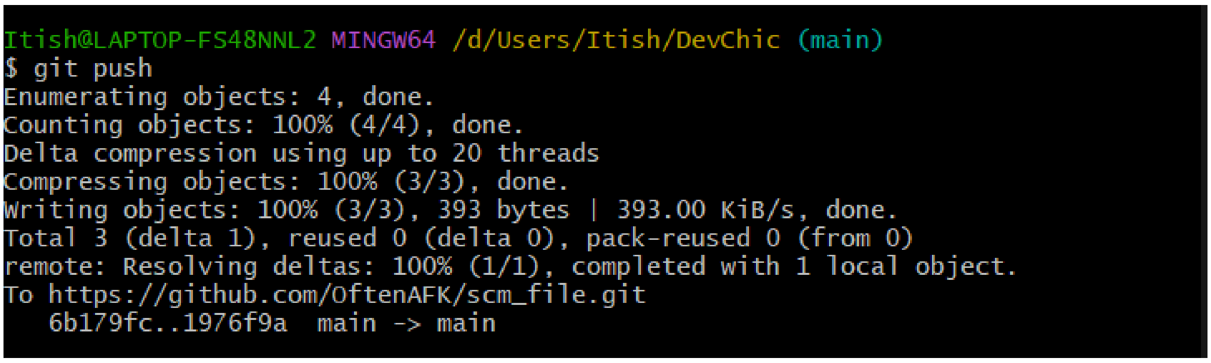
```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git remote add origin https://github.com/OftenAFK/scm_file
error: remote origin already exists.

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git remote -v
origin  https://github.com/OftenAFK/scm_file.git (fetch)
origin  https://github.com/OftenAFK/scm_file.git (push)
```

- The command "git remote add origin <link_of_repository>" is used to initiate and upstream of data that is to be transmitted to GitHub.
- The command "git remote -v" provides the list of remote repositories associated with our local git repository. Also used for verifying the URLs of your remotes before pushing or pulling.

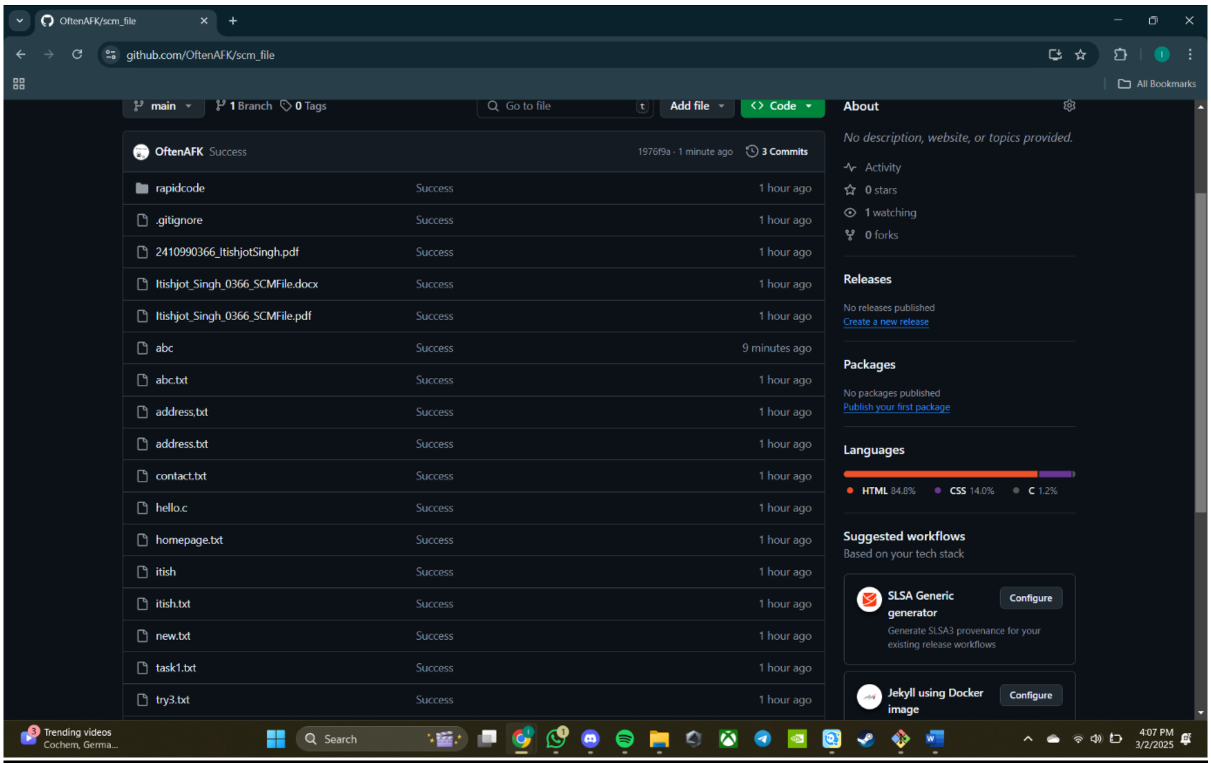


```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 301 bytes | 301.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/OftenAFK/scm_file.git
   a66417d..6b179fc  main -> main
branch 'main' set up to track 'origin/main'.
```

- The command "git push -u origin main" pushes the data from local repository to the GitHub online repository.
- Using "git push -u origin main" sets the upstream branch for main, making the future git push and git pull commands to be able to function without specifying the remote and branch.



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 393 bytes | 393.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/OftenAFK/scm_file.git
   6b179fc..1976f9a  main -> main
```



github.com/OftenAFK/scm_file

main · 1 Branch · 0 Tags

OftenAFK Success — 1976f9a · 1 minute ago — 3 Commits

rapidcode	Success	1 hour ago
.gitignore	Success	1 hour ago
2410990366_ItishjotSingh.pdf	Success	1 hour ago
Itishjot_Singh_0366_SCMFile.docx	Success	1 hour ago
Itishjot_Singh_0366_SCMFile.pdf	Success	1 hour ago
abc	Success	9 minutes ago
abc.txt	Success	1 hour ago
address,txt	Success	1 hour ago
address.txt	Success	1 hour ago
contact.txt	Success	1 hour ago
hello.c	Success	1 hour ago
homepage.txt	Success	1 hour ago
itish	Success	1 hour ago
itish.txt	Success	1 hour ago
new.txt	Success	1 hour ago
task1.txt	Success	1 hour ago
try3.txt	Success	1 hour ago

About

No description, website, or topics provided.

Activity · 0 stars · 1 watching · 0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Languages

HTML 84.8% · CSS 14.0% · C 1.2%

Suggested workflows

Based on your tech stack

SLSA Generic generator — Generate SLSA3 provenance for your existing release workflows

Jekyll using Docker image

- The file is now available in the online GitHub repository

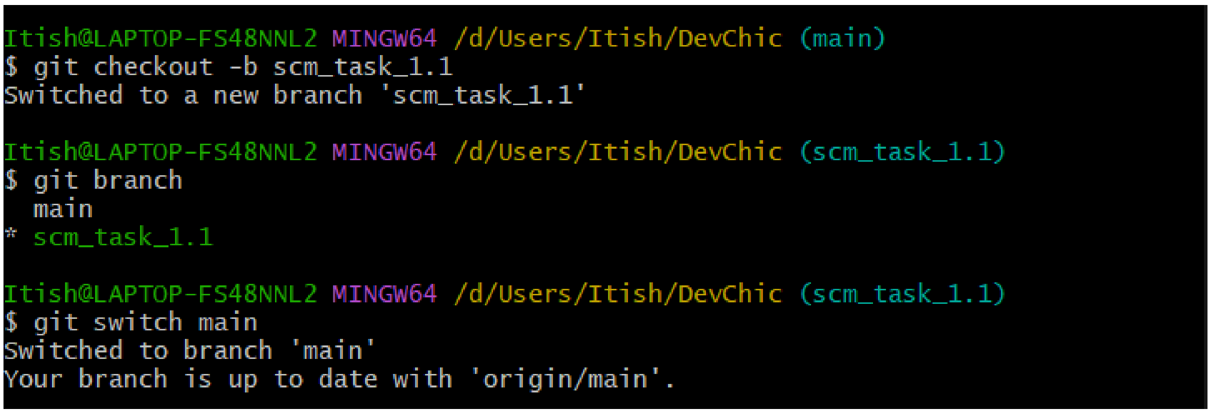
Practical 3

Aim: To merge two branches within a Git repository**.**

Theory:
Merging branches in Git allows you to combine changes from one branch into another. It is a fundamental process in collaborative workflows, ensuring all contributions are integrated into a single codebase.

Procedure:

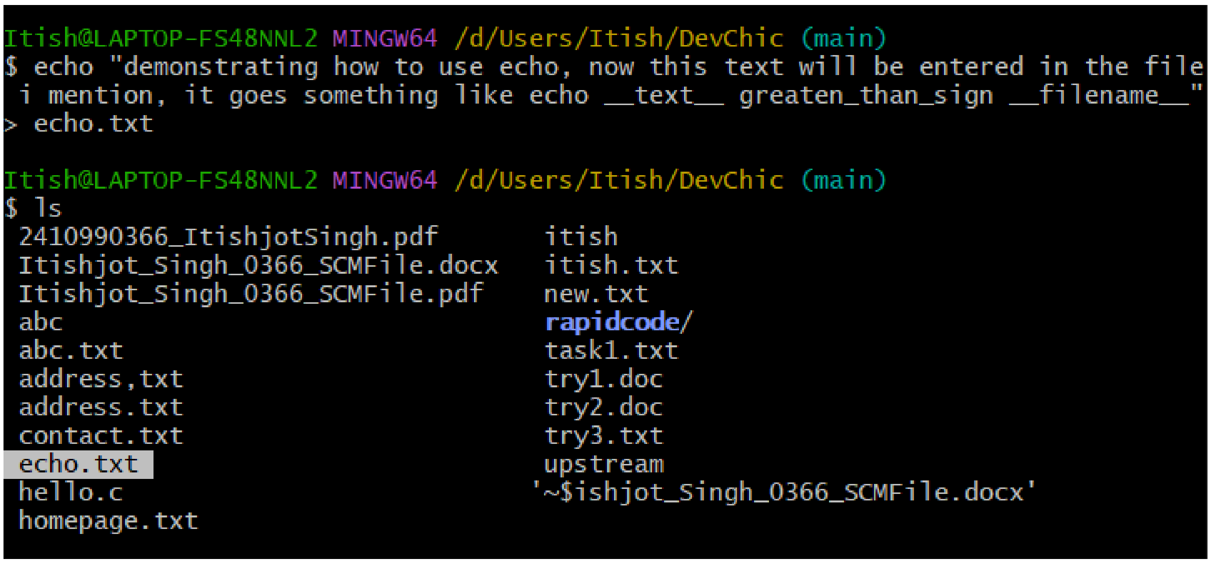
1. Create a new branch and switch to it:



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git checkout -b scm_task_1.1
Switched to a new branch 'scm_task_1.1'

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (scm_task_1.1)
$ git branch
  main
* scm_task_1.1

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (scm_task_1.1)
$ git switch main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
```

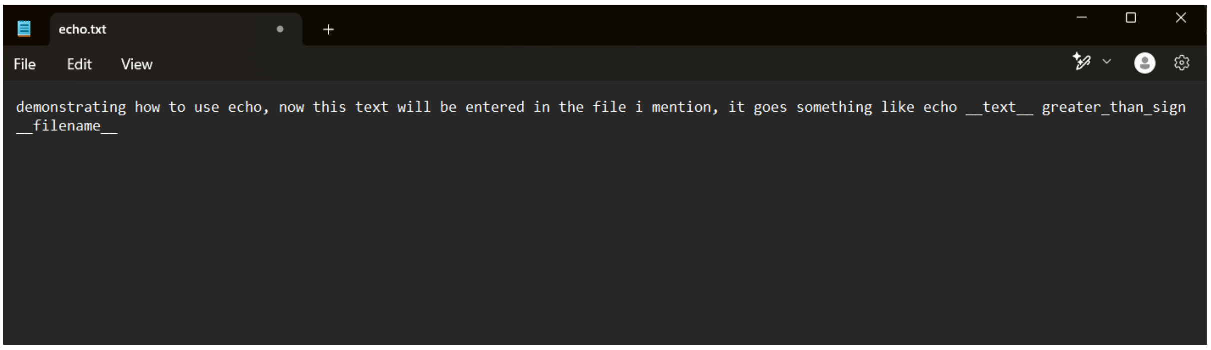
- The command "git checkout -b <branch_name>" makes a new branch and automatically switches to it.
- Whereas, commands like "git switch <branch_name>" and "git checkout <branch_name>" are used to switch between already existing branches.

2. Make changes to a file in the new branch and commit them:

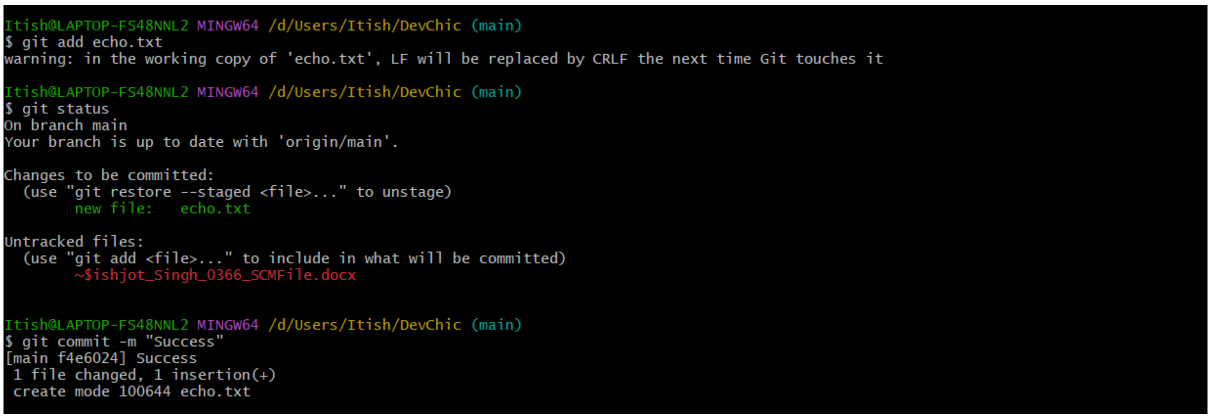


```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ echo "demonstrating how to use echo, now this text will be entered in the file
 i mention, it goes something like echo __text__ greaten_than_sign __filename__"
> echo.txt

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ ls
 2410990366_ItishjotSingh.pdf       itish
 Itishjot_Singh_0366_SCMFile.docx   itish.txt
 Itishjot_Singh_0366_SCMFile.pdf    new.txt
 abc                                rapidcode/
 abc.txt                            task1.txt
 address,txt                        try1.doc
 address.txt                        try2.doc
 contact.txt                        try3.txt
 echo.txt                           upstream
 hello.c                           '~$ishjot_Singh_0366_SCMFile.docx'
 homepage.txt
```

- Using the command “echo -text- > -filename- creates a new file and adds the text that the user typed.



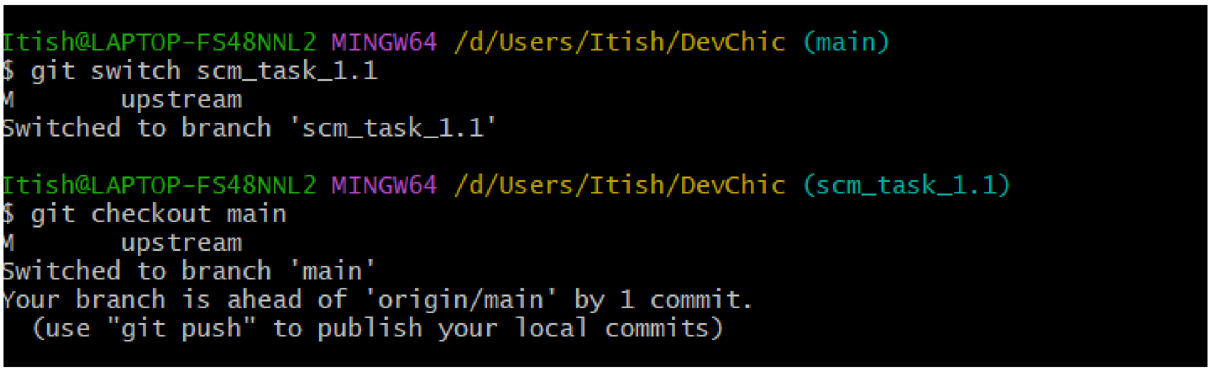
```
demonstrating how to use echo, now this text will be entered in the file i mention, it goes something like echo __text__ greater_than_sign
__filename__
```



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git add echo.txt
warning: in the working copy of 'echo.txt', LF will be replaced by CRLF the next time Git touches it

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (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)
        new file:   echo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ~$ishjot_Singh_0366_SCMFile.docx


Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git commit -m "Success"
[main f4e6024] Success
 1 file changed, 1 insertion(+)
 create mode 100644 echo.txt
```

- The new file named echo.txt has now been committed.

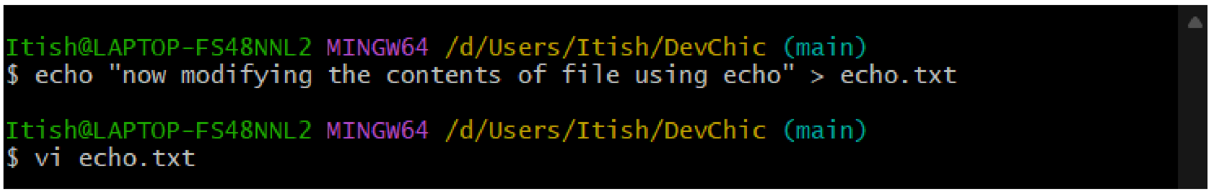
3. Switch back to the main branch:



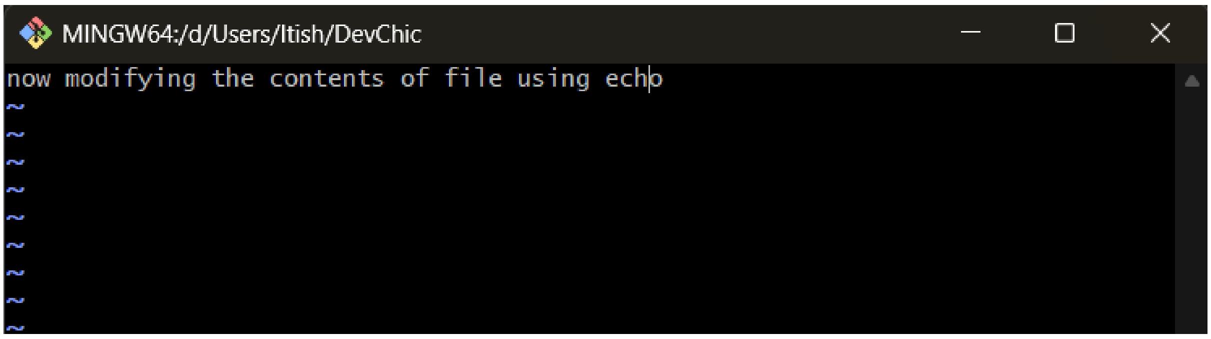
```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git switch scm_task_1.1
M       upstream
Switched to branch 'scm_task_1.1'

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (scm_task_1.1)
$ git checkout main
M       upstream
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
```

- Using commands like git switch and git checkout to switch between main and the other branches.

4. Modify another file in the main branch and commit the changes:

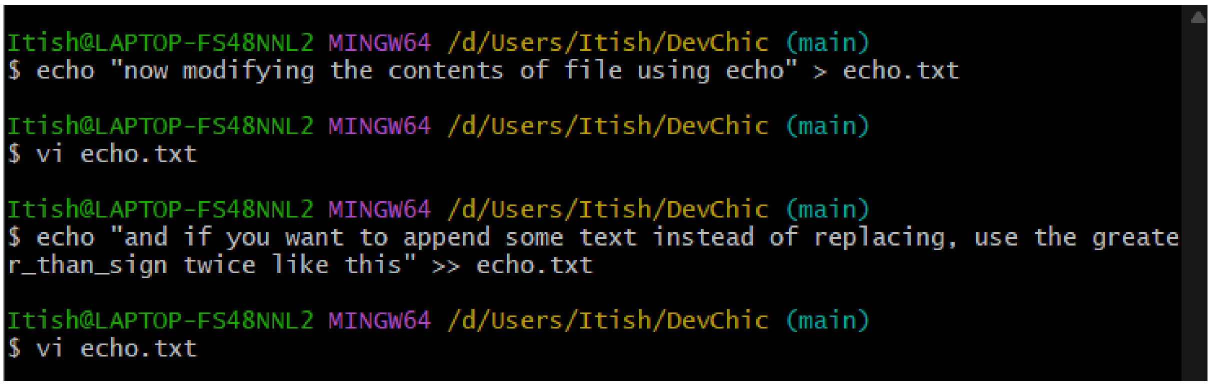


```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ echo "now modifying the contents of file using echo" > echo.txt

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ vi echo.txt
```



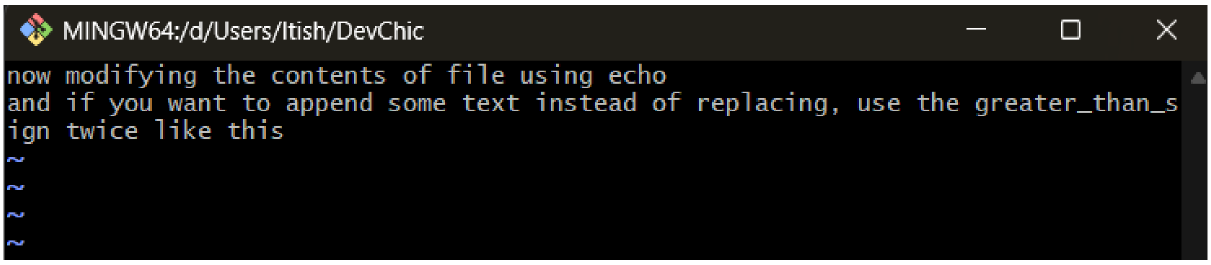
```
now modifying the contents of file using echo
~
~
~
~
~
~
~
~
~
```

- The contents of the file “echo.txt” have now been modified.



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ echo "now modifying the contents of file using echo" > echo.txt

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ vi echo.txt

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ echo "and if you want to append some text instead of replacing, use the greate
r_than_sign twice like this" >> echo.txt

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ vi echo.txt
```

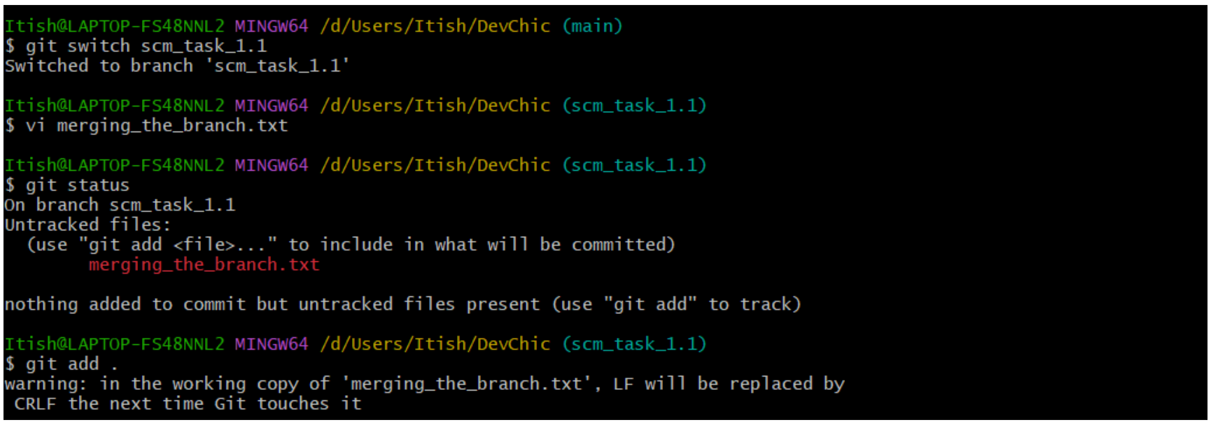
- While editing is a way to do things, appending is an option as well. While appending, instead of using the greater than sign (i.e. “>”) just once, we use twice.



```
now modifying the contents of file using echo
and if you want to append some text instead of replacing, use the greater_than_s
ign twice like this
~
~
~
~
```

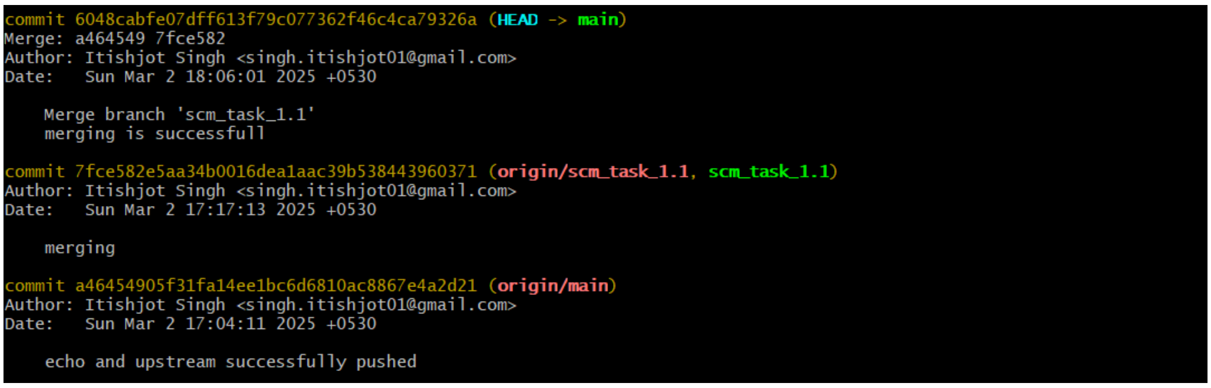
- The contents of echo.txt have now been edited and well as appended by another text.

5. Merge the new branch into the main branch:



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git switch scm_task_1.1
Switched to branch 'scm_task_1.1'

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (scm_task_1.1)
$ vi merging_the_branch.txt

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (scm_task_1.1)
$ git status
On branch scm_task_1.1
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        merging_the_branch.txt

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

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (scm_task_1.1)
$ git add .
warning: in the working copy of 'merging_the_branch.txt', LF will be replaced by
 CRLF the next time Git touches it
```

- Adding more files to the secondary branch so that it can be merged with main.



```
commit 6048cabfe07dff613f79c077362f46c4ca79326a (HEAD -> main)
Merge: a464549 7fce582
Author: Itishjot Singh <singh.itishjot01@gmail.com>
Date:   Sun Mar 2 18:06:01 2025 +0530

    Merge branch 'scm_task_1.1'
    merging is successfull

commit 7fce582e5aa34b0016dea1aac39b538443960371 (origin/scm_task_1.1, scm_task_1.1)
Author: Itishjot Singh <singh.itishjot01@gmail.com>
Date:   Sun Mar 2 17:17:13 2025 +0530

    merging

commit a46454905f31fa14ee1bc6d6810ac8867e4a2d21 (origin/main)
Author: Itishjot Singh <singh.itishjot01@gmail.com>
Date:   Sun Mar 2 17:04:11 2025 +0530

    echo and upstream successfully pushed
```

- Using the command "git log" we can configure that the branch has successfully been merged with main.

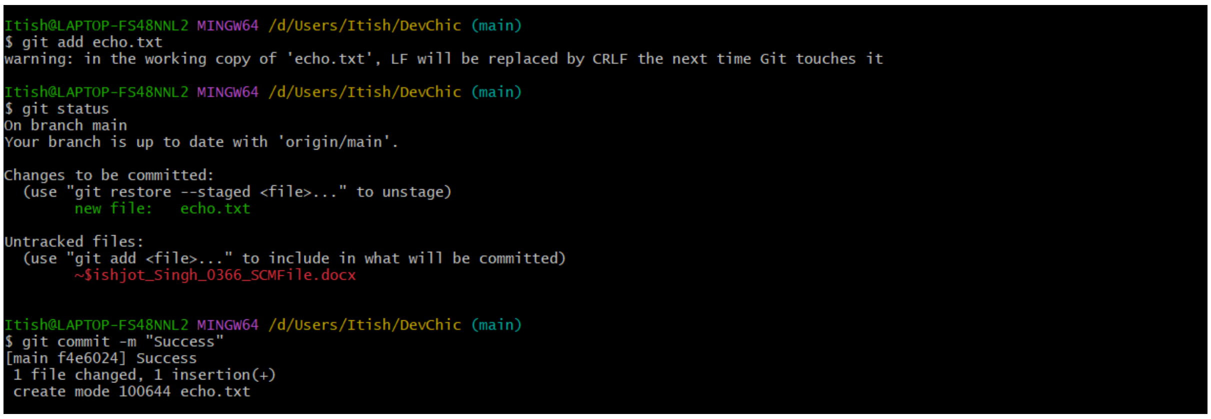
Practical 4

Aim:
To demonstrate push and pull operations in Git.

Theory:
Push transfers committed changes from the local repository to the remote repository, while pull retrieves updates from the remote repository.

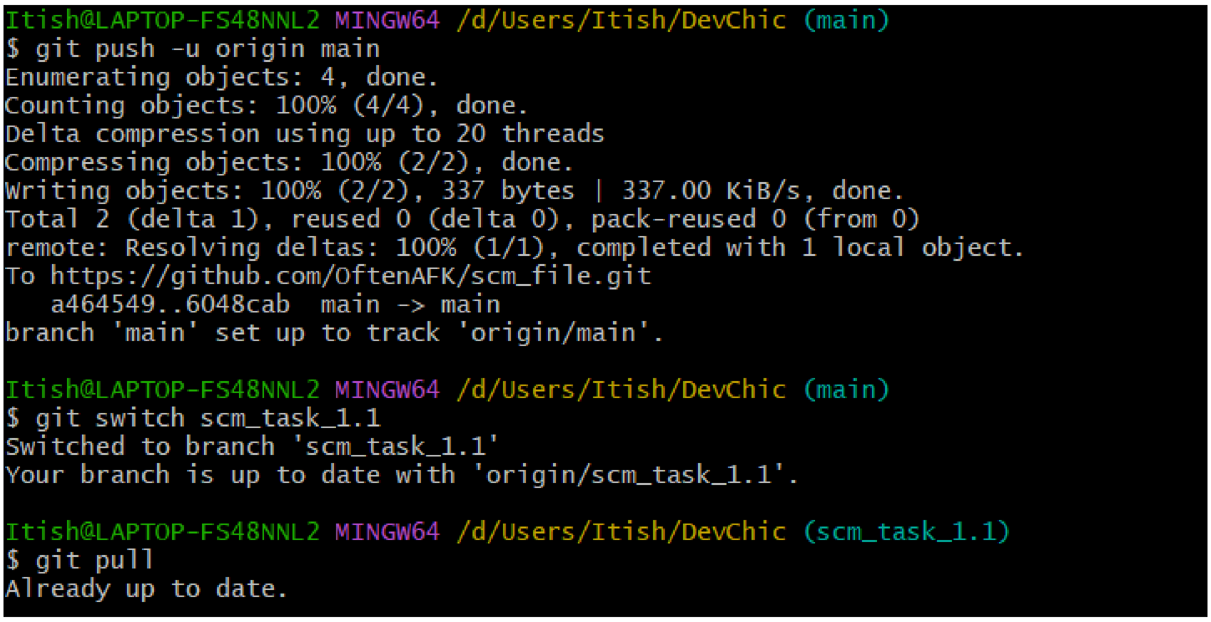
Procedure:
○ Make changes in the local repository and commit them.
○ Push the changes to the remote repository using git push.
○ Make changes directly on the remote repository (e.g., via GitHub interface).
○ Pull the changes to the local repository using git pull.

Tasks:
Provide screenshots of the push and pull operations.
Include the updated commit log.



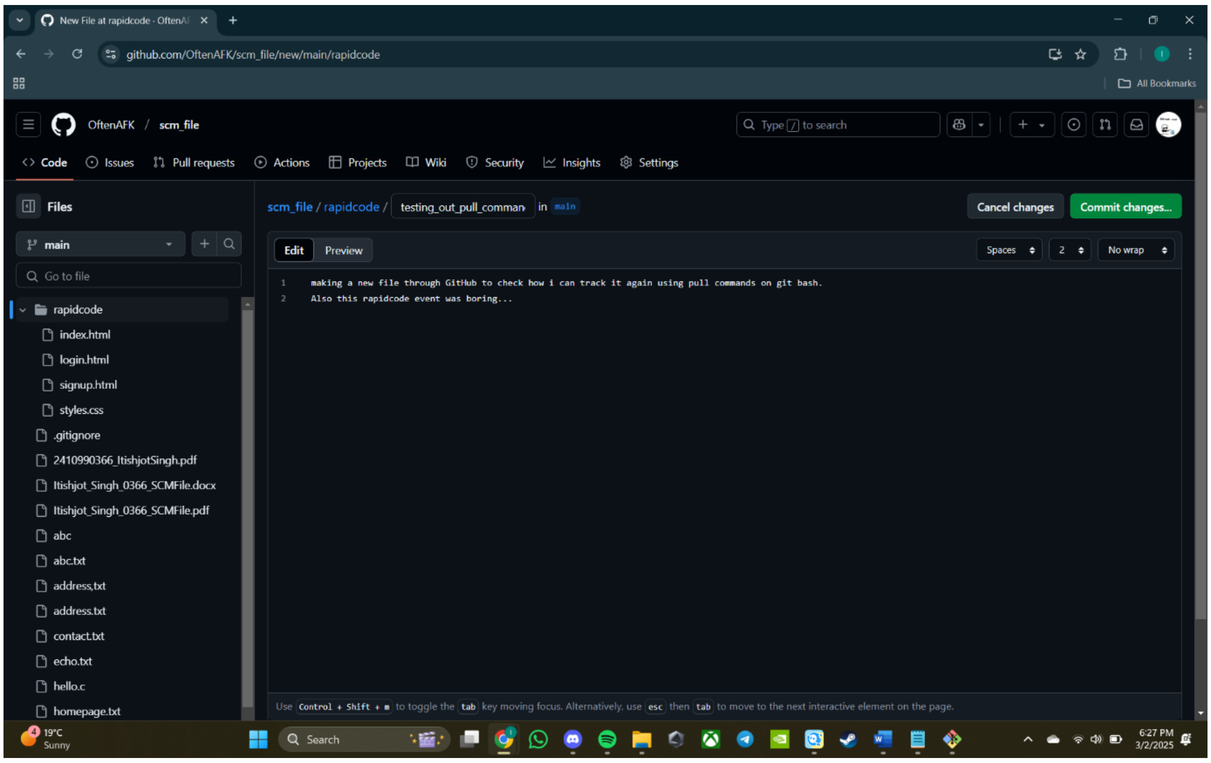
```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git add echo.txt
warning: in the working copy of 'echo.txt', LF will be replaced by CRLF the next time Git touches it

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (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)
        new file:   echo.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        ~$ishjot_Singh_0366_SCMFile.docx


Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git commit -m "Success"
[main f4e6024] Success
 1 file changed, 1 insertion(+)
 create mode 100644 echo.txt
```

○ The file echo.txt was edited once again and after editing, it was pushed to the GitHub repository.



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git push -u origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 20 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 337 bytes | 337.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/OftenAFK/scm_file.git
   a464549..6048cab  main -> main
branch 'main' set up to track 'origin/main'.

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (main)
$ git switch scm_task_1.1
Switched to branch 'scm_task_1.1'
Your branch is up to date with 'origin/scm_task_1.1'.

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic (scm_task_1.1)
$ git pull
Already up to date.
```

- After using push and pull commands, we are up to date with all the files in the repository.
- Git pull is majorly used to detect what changes have been made by the user as well as other collaborators working in that GitHub repository, since its just the user making changes here, everything is up to date.
- Though git bash won’t detect if the changes are made directly through the GUI of GitHub, and to update the git bash about the changes that were made on GitHub, we use git pull.



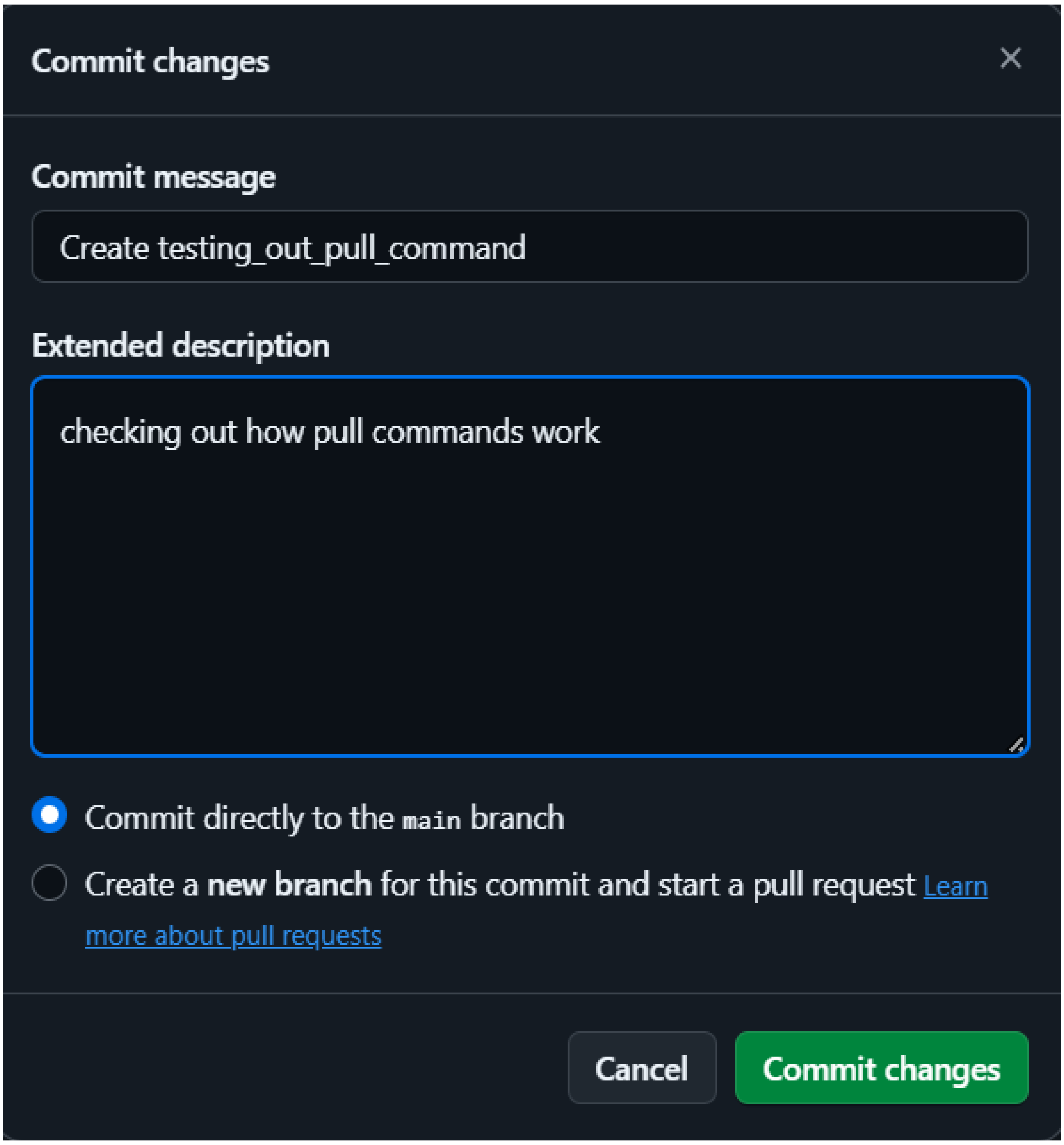
github.com/OftenAFK/scm_file/new/main/rapidcode

OftenAFK / scm_file

scm_file / rapidcode / testing_out_pull_comman in main

```
making a new file through GitHub to check how i can track it again using pull commands on git bash.
Also this rapidcode event was boring...
```

- Making a new file under the rapidcode directory, the file “testing_out_pull_command” was added through GitHub.



Commit changes

Commit message

Create testing_out_pull_command

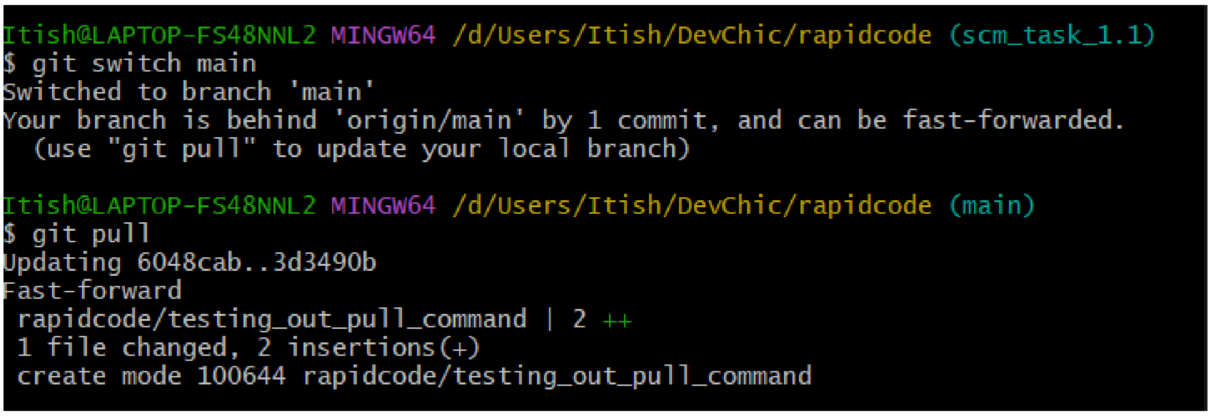
Extended description

checking out how pull commands work

[✓ Commit directly to the `main` branch]

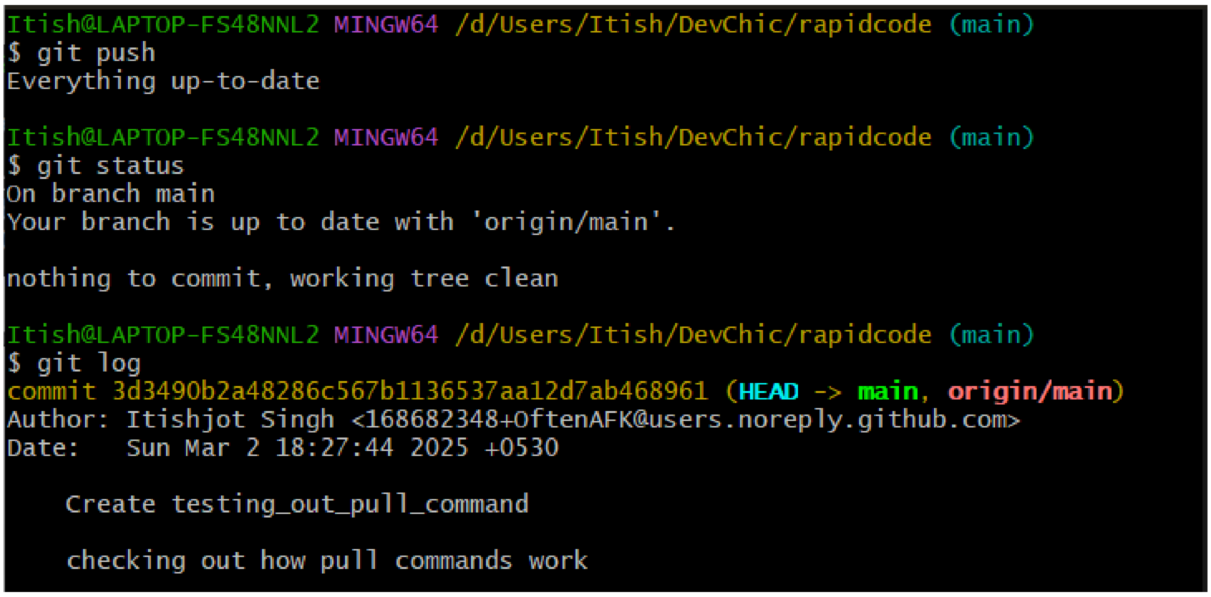
Create a **new branch** for this commit and start a pull request Learn more about pull requests

- The file “testing_out_pull_command” now being committed through GitHub.



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic/rapidcode (scm_task_1.1)
$ git switch main
Switched to branch 'main'
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic/rapidcode (main)
$ git pull
Updating 6048cab..3d3490b
Fast-forward
 rapidcode/testing_out_pull_command | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 rapidcode/testing_out_pull_command
```

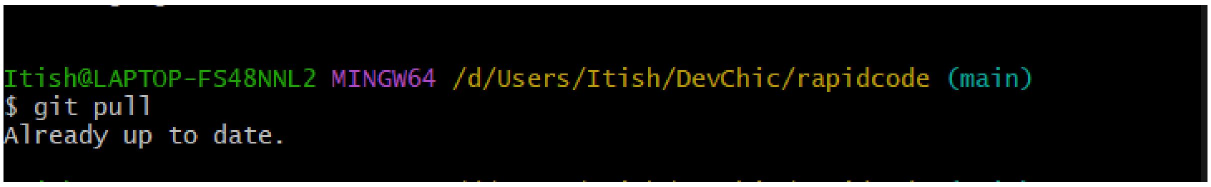
- Now using the command "git pull" we can configure that a change has been made in the rapidcode directory.



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic/rapidcode (main)
$ git push
Everything up-to-date

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic/rapidcode (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic/rapidcode (main)
$ git log
commit 3d3490b2a48286c567b1136537aa12d7ab468961 (HEAD -> main, origin/main)
Author: Itishjot Singh <168682348+OftenAFK@users.noreply.github.com>
Date:   Sun Mar 2 18:27:44 2025 +0530

    Create testing_out_pull_command

    checking out how pull commands work
```

- The working tree is clean and we have nothing to commit, hence we are good to go. Now using git log we can check the history and see that git bash is now up to date with the change we made at our GitHub repository.

<u>Therefore:</u>



```
Itish@LAPTOP-FS48NNL2 MINGW64 /d/Users/Itish/DevChic/rapidcode (main)
$ git pull
Already up to date.
```