

What is Git?

Git is a distributed version control system which tracks changes in computer files, primarily used for coordinating development work by the programmers during software development process. It helps developers track changes in code, collaborate, and manage project history.

What does Git do?

- Manage projects with Repositories
- Clone a project to work on a local copy
- Control and track changes with Staging and Committing
- Branch and Merge to allow for work on different parts and versions of a project
- Pull the latest version of the project to a local copy
- Push local updates to the main project

Working with Git

- Initialize Git on a folder, making it a Repository
- Git now creates a hidden folder to keep track of changes in that folder
- When a file is changed, added or deleted, it is considered modified
- You select the modified files you want to Stage
- The Staged files are Committed, which prompts Git to store a permanent snapshot of the files
- Git allows you to see the full history of every commit.
- You can revert back to any previous commit.
- Git does not store a separate copy of every file in every commit, but keeps track of changes made in each commit!

What is GitHub?

GitHub is a cloud-based platform for hosting and sharing Git repositories, enabling collaboration on code.

- Git is not the same as GitHub.
- GitHub makes tools that use Git.
- GitHub is the largest host of source code in the world, and has been owned by Microsoft since 2018.

I. Installing Git

- **Windows:** [Download Git](#)
- **macOS:** brew install git
- **Linux:** sudo apt install git


II. Using Git with Command Line

To start using Git, we are first going to open up our Command shell.

For Windows, you can use Git bash, which comes included in Git for Windows. For Mac and Linux you can use the built-in terminal.

The first thing we need to do, is to check if Git is properly installed:

- If Git is installed, it should show something like git version X.Y
`$ git --version`

 MINGW64:/c:/Users/nirup/Desktop/Git tutorial

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ git --version
git version 2.49.0.windows.1
```

III. Initial Configuration (only once)

Now let Git know who you are. This is important for version control systems, as each Git commit uses this information:

- It sets name for all Git commits on this system.
`$ git config --global user.name "Nirupom Saha"`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ git config --global user.name "Nirupom Saha"

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ |
```

- It sets email for commit identification.

`$ git config --global user.email "nirupomsaha1234@gmail.com"`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ git config --global user.email "nirupomsaha1234@gmail.com"

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ |
```

- It shows all global Git configuration settings.

`$ git config --global --list`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ git config --global --list
user.email=nirupomsaha1234@gmail.com
user.name=Nirupom Saha
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ |
```

IV. Create a New Repository

- **mkdir** makes a new directory.

`$ mkdir project`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ mkdir project

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ |
```

- **cd** changes the current working directory.

`$ cd project`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial
$ cd project

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project
$ |
```

V. Initialize a Git Repository

- This initializes the directory as a Git repository by creating a .git subdirectory.

`$ git init`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project
$ git init
Initialized empty Git repository in C:/Users/nirup/Desktop/Git tutorial/project/.git/

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ |
```

VI. Git Workflow (Local)

Working Directory → Staging Area → Local Repository → Remote Repository (GitHub)

VII. Check Repository Status

- It displays the current status of the repository, showing which files are staged, modified, or untracked.

`$ git status`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git status
On branch master

No commits yet

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

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ |
```

VIII. Create a file

`$ touch file1.txt`

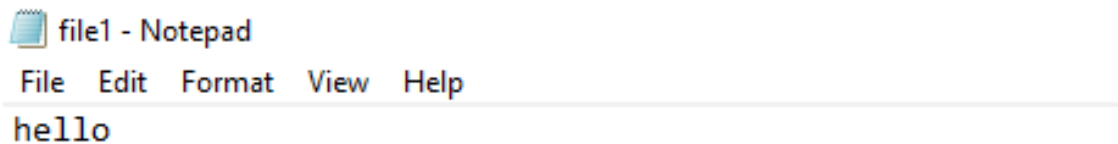
```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ touch file1.txt

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$
```

`$ echo hello >> file1.txt`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ echo hello >> file1.txt

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$
```



IX. Add Files to Staging

- Stages a specific file to be committed. Only staged files are included in the next commit.

`$ git add file1.txt`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git add file1.txt
warning: in the working copy of 'file1.txt', LF will be replaced by CRLF the next time Git touches it
```

- Adds all changes (new, modified, deleted files) to staging.

`$ git add .`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git add .
```

X. Commit Changes

- Saves the staged changes to the local repository with a message describing what was done.

`$ git commit -m "Updated file1.txt"`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git commit -m "Updated file1.txt"
[master (root-commit) f06ecb0] Updated file1.txt
1 file changed, 1 insertion(+)
create mode 100644 file1.txt
```

XI. Viewing History and Changes

- Shows full history of commits (with author, date, message, commit ID).

`$ git log`

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git log
commit f06ecb0d29e52e34cb0bbff8789366a98c203fb4 (HEAD -> master)
Author: Nirupom Saha <nirupomsaha1234@gmail.com>
Date: Wed Apr 9 01:48:39 2025 +0530

    Updated file1.txt
```

- Shows a condensed version of log (one line per commit).

```
$ git log --oneline
```

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git log --oneline
f06ecb0 (HEAD -> master) Updated file1.txt

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$
```

XII. Connect to GitHub

- Create a repository on GitHub.
- Links your local Git repo to a GitHub repository.

```
$ git add origin https://github.com/username/my-project.git
```

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git remote add origin https://github.com/Nirupom-Saha/Wipro.git

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$
```

- Renames your default branch to main (optional, GitHub uses main now)

```
$ git branch -M main
```

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (master)
$ git branch -M main

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$
```

- Pushes your local code to GitHub and sets it as the default upstream.

```
$ git push -u origin main
```

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git push -u origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 720 bytes | 360.00 KiB/s, done.
Total 7 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To https://github.com/Nirupom-Saha/Wipro.git
    bd89bfc..3571c0b  main -> main
branch 'main' set up to track 'origin/main'.

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$
```

XIII. Cloning a Repository (Get from GitHub)

- Downloads the project with full version control history.
\$ git clone <https://github.com/username/repo-name.git>

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git clone https://github.com/Nirupom-Saha/Wipro.git
Cloning into 'Wipro'...
remote: Enumerating objects: 37, done.
remote: Counting objects: 100% (37/37), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 37 (delta 12), reused 8 (delta 2), pack-reused 0 (from 0)
Receiving objects: 100% (37/37), 162.70 KiB | 293.00 KiB/s, done.
Resolving deltas: 100% (12/12), done.
```

XIV. Pulling and Pushing Changes

- Fetches and merges changes from the GitHub repo into your local branch.
\$ git pull origin main

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git pull origin main
From https://github.com/Nirupom-Saha/Wipro
 * branch          main          -> FETCH_HEAD
Already up to date.

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$
```

- Sends your local commits to GitHub.
\$ git push origin main

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git push origin main
Everything up-to-date

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ |
```

XV. Branching (Parallel Development)

- Shows all local branches.
\$ git branch

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git branch
* main

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$
```

- It will create a new branch called **feature-branch**.

\$ git branch feature-branch

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git branch feature-branch
```

- It will switch to the branch **feature-branch**

\$ git checkout feature-branch

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git checkout feature-branch
Switched to branch 'feature-branch'

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (feature-branch)
$
```

- It will create and switch to a new branch in one command.

\$ git checkout -b feature-auth

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (feature-branch)
$ git checkout -b feature-auth
Switched to a new branch 'feature-auth'

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (feature-auth)
$ |
```

- Merges feature-login branch into the current branch (often main)

\$ git merge feature-branch

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git merge feature-branch
Already up to date.
```

- Deletes the local branch after merging.

\$ git branch -d feature-auth

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ git branch -d feature-auth
Deleted branch feature-auth (was 999434e).
```

XVI. Undo Changes

- It will discard local changes in a file (not staged or committed).

\$ git checkout -- filename.txt

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git checkout -- file1.txt
```


- It will unstage the file (removes it from the staging area).

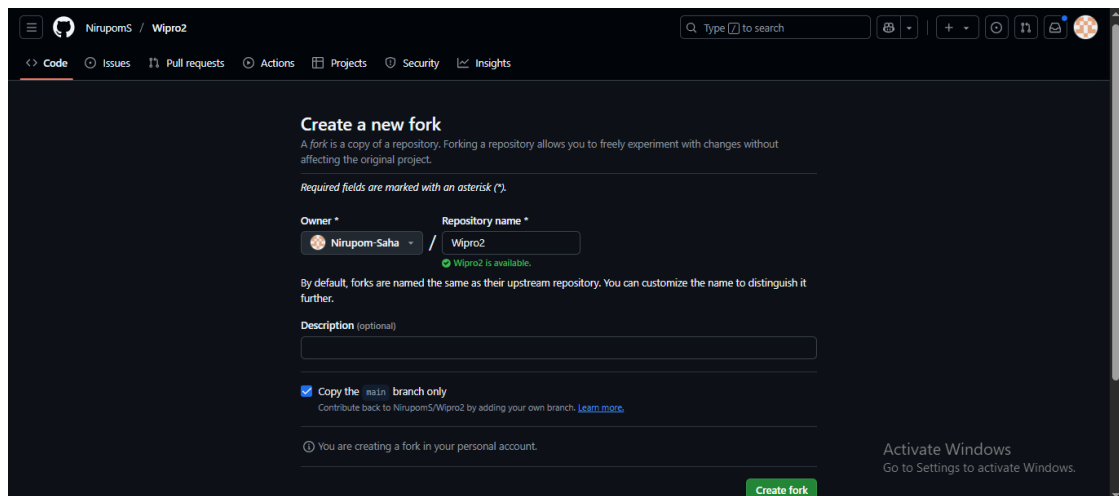
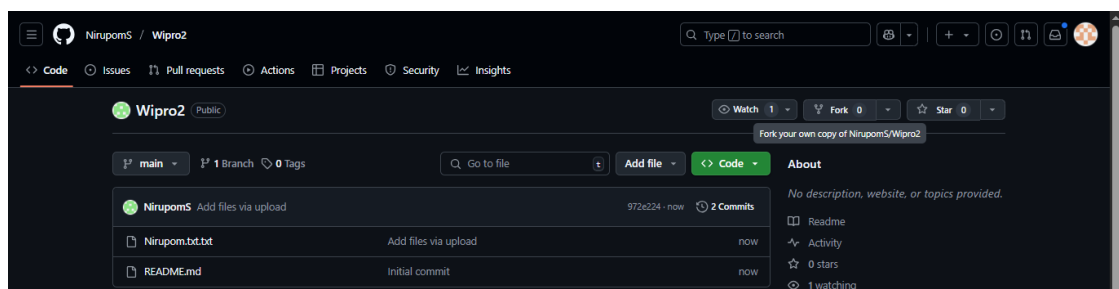
```
$ git reset filename.txt
```

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$ git reset file1.txt

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/Git tutorial/project (main)
$
```

XVII. Contributing to Open Source (GitHub Flow)

- **Fork** the repo on GitHub



- **Clone** it to your machine

```
MINGW64:/c/Users/nirup/Desktop/ASS3
```

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3
$ git clone https://github.com/Nirupom-Saha/wipro2.git
Cloning into 'wipro2'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (6/6), done.

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3
$
```

- **Create a branch** for your feature

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (main)
$ git checkout -b my-new-feature
Switched to a new branch 'my-new-feature'

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$
```

- Make changes, commit, push

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ echo "This is a new feature" > feature.txt

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$
```

```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ git add feature.txt
warning: in the working copy of 'feature.txt', LF will be replaced by CRLF the next time Git touches it

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ |
```

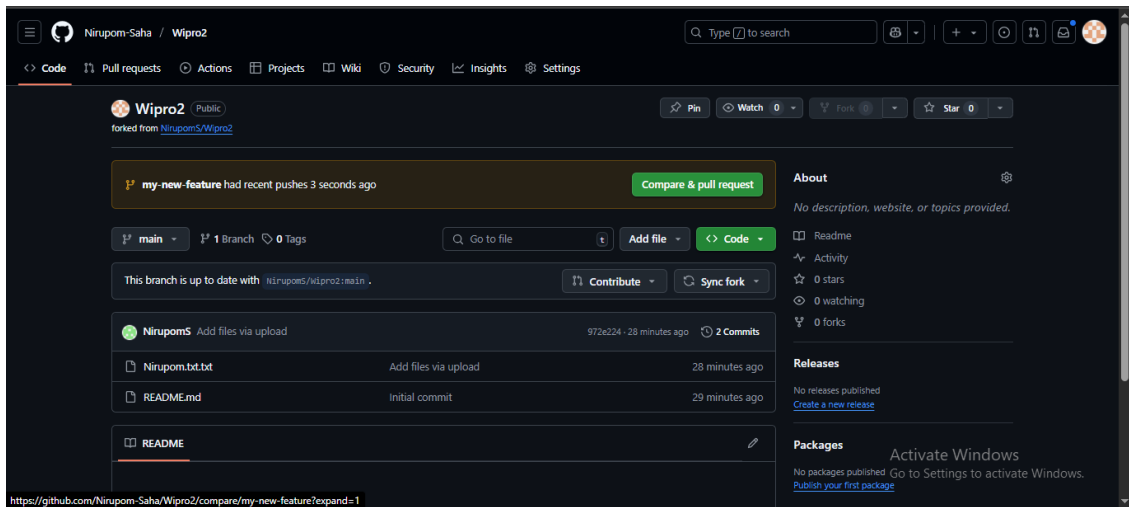
```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ git commit -m "Add new feature"
[my-new-feature 999434e] Add new feature
1 file changed, 1 insertion(+)
create mode 100644 feature.txt

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ |
```

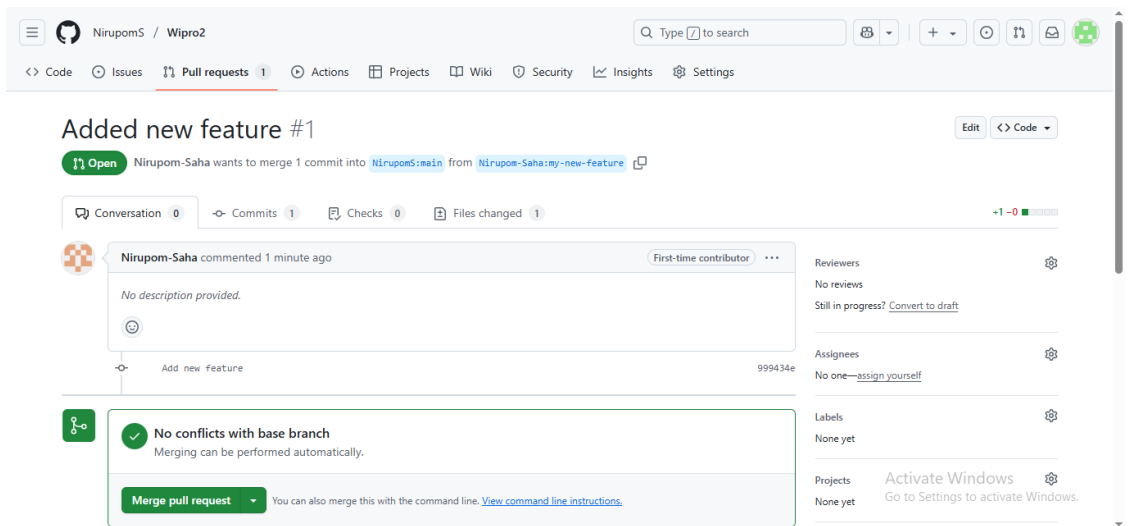
```
nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ git push origin my-new-feature
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 343 bytes | 171.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'my-new-feature' on GitHub by visiting:
remote:   https://github.com/Nirupom-Saha/Wipro2/pull/new/my-new-feature
remote:
To https://github.com/Nirupom-Saha/Wipro2.git
 * [new branch]      my-new-feature -> my-new-feature

nirup@DESKTOP-OBQE6SK MINGW64 ~/Desktop/ASS3 (my-new-feature)
$ |
```

- **Open a Pull Request (PR)**



- **Merge a Pull Request**



XVIII. Helpful Git Commands

Command

git stash

git stash pop

git remote -v

git branch -a

git fetch

git rebase branchname

git tag v1.0

What it does

Temporarily saves uncommitted changes

Re-applies stashed changes

Shows remote repository URLs

Lists local and remote branches

Downloads latest changes (without merging)

Applies commits on top of another branch

Tags a specific commit (useful for releases)