

# Git

**Name : Vaishak E Bhandary**

**MGSA\_091**

## **Introduction :**

### **Version Control:**

Version Control systems is a software tool that helps to track the files or set of data over time so that if one needs, then he can recall back to the specific version later. It helps to manage your codebase very efficiently. If a mistake appears, then developers can easily compare the earlier versions of code and fix the problem and manage the disruption between the team members.

There are undoubtedly different types of version control, but here I am going to talk about Git specifically.

### **Git and Github:**

Git is by far the most widely used modern version control system. It is an actively maintained open source project which was initially developed by **Linus Torvalds** in 2005, the creator of Linux operating system kernel.

Github is a code hosting platform that is based on Git version control for collaborations and like-minded people.

Before moving further, there are some terms one should know,

- **Repository:-** A GitHub repository is used to store the development of the project. It also includes a license file and a README file about the project.
- **Branch:-** A GitHub branch enables developers to create and work on different versions of the repository simultaneously. By default, a repository contains a master branch, and other branches are just a copy of it.
- **Commits:-** Changes made are recorded.
- **Pull requests:-** It helps collaborators to merge the proposed changes with the master branch.

## Git Commands :

1. **git init** - When you first start any project, type this in **git bash** or terminal, it initialize the git repository in your folder.
2. **touch .gitignore** - You dont want to keep everything in source control, especially all the heavy dependencies file. **Note:-** 'touch' only works when you are in your project directory
3. **git add .** - It stages all the changes in the files for commit.
4. **git reset .** - It unstaged the file from a commit.
5. **git reset --hard** - It not only unstaged the file but delete it too, so be careful in using this command. **Note:-** Make a habit for small commits; even you change one or two lines of code, commit it so in future it would be easy to track the faulty codes.
6. **git commit -m "\$your\_message"** - After staging all your changes, it commits your changes with a message so that you can remember at a certain point what did you change. **Note:-** To get GitHub's star start using emojis during committing because more stars mean how good a project is.
7. **git branch** - It tells in which branch you are working in the repository.
8. **git checkout -b "\$branch\_name"** - It helps you to create a new branch with a given branch name.
9. **git stash -u** - If you don't want to commit your progress but save it for future use, then stash command takes your modified tracked files, stage the changes, save them on a stack of unfinished changes to reapply at any time.
10. **git stash pop** - Then later, if you want to apply the committed stack, you can use this command.
11. **git merge "\$branch\_name"** - It let you merge your other branch with the master branch, your branch, and master branch id becomes the same. **Note:-** This sometimes leads to conflicts. Then you have to review it and fix the conflicts manually; this why committing to small changes are considered the best choice.
12. **git merge "\$branch\_name" --squash** - Sometimes commits in child branches are ahead of commits of the master branch so to keep the commit history concise, this command helps to squash them down to single commit but still preserve all the commits in child branches.

13. **git remote add origin "\$url"** - It helps to add a new remote in the directory your repository is stored at. **Note:-** It takes two arguments, remote name "origin" and second remote URL.
14. **git push -u origin master** - this only pushes your master branch to origin.

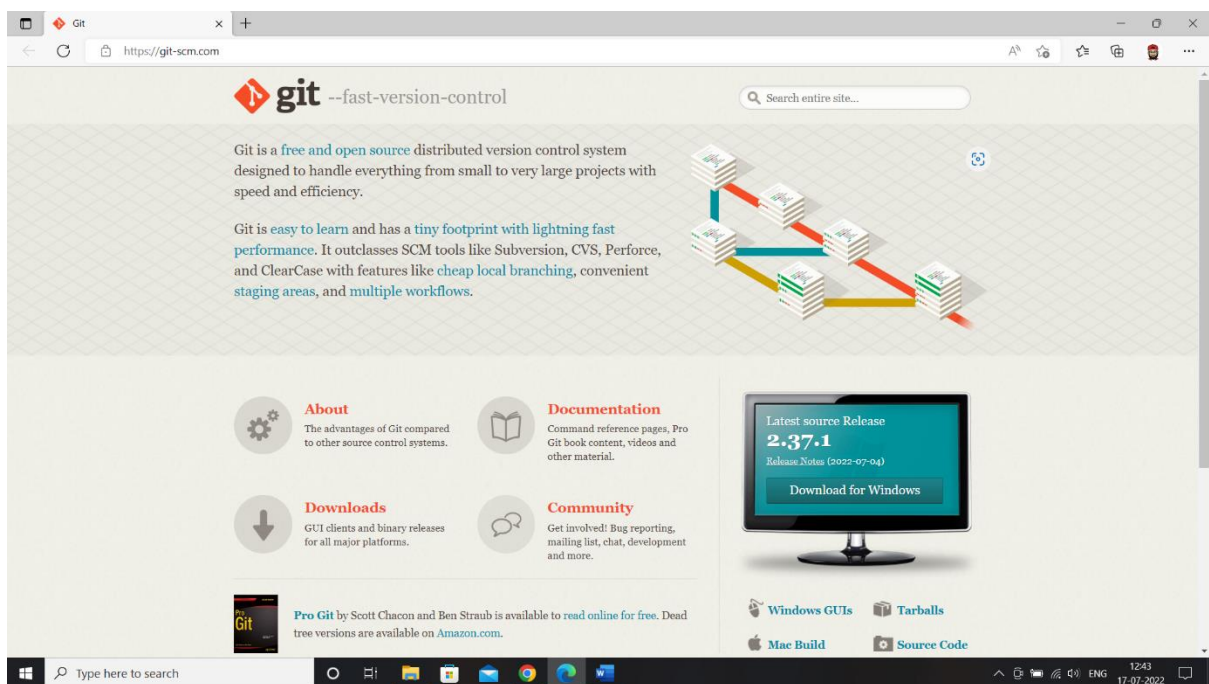
## Git bash :

**Git Bash** is a source control management system for Windows. It allows users to type Git commands that make source code management easier through versioning and commit history. **Bash** is a Linux-based command line (that has been ported over to Windows) while **Shell** is a native Windows command line.

Installing Git bash :

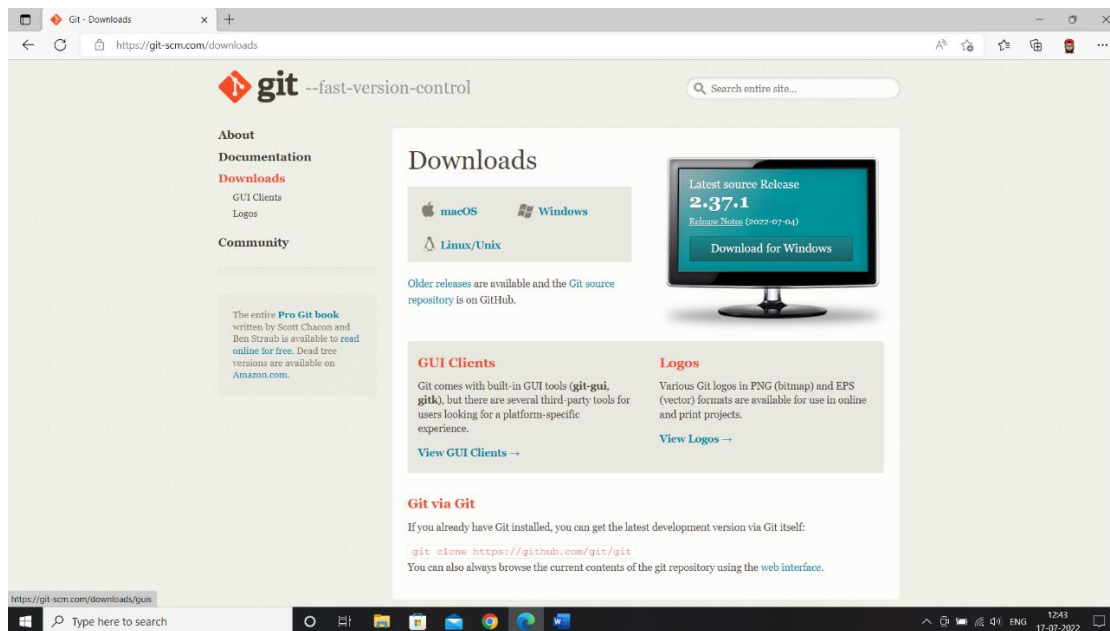
Step 1 :

Download the Git Bash setup from the official website: <https://git-scm.com/>



## Step 2

Download the installer.



## Step 3

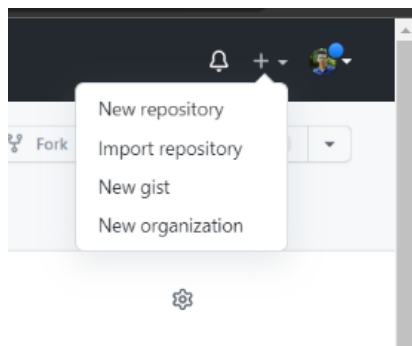
Run the .exe file you just downloaded and follow the instructions in the installer.

## Step 4

Run Git Bash by right-clicking on any folder and selecting the **Git Bash Here** option from the context menu(right-click menu).

## Git Terminal :

Step 1 : Create a new repository on GitHub.com. To avoid errors, do not initialize the new repository with README, license, or gitignore files. You can add these files after your project has been pushed to GitHub.



github.com/new

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?  
[Import a repository.](#)

Owner <sup>\*</sup>

Repository name <sup>\*</sup>

vaishakbhandary / Demo

Great repository names are short Demo is available. Need inspiration? How about [turbo-couscous](#)?

Description (optional)

☒ Public

Anyone on the internet can see this repository. You choose who can commit.

☐ Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

model.drawio.png Show all

Type here to search

github.com/vaishakbhandary/Demo

Search or jump to...

Pull requests Issues Marketplace Explore

vaishakbhandary / Demo Public

Pin Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

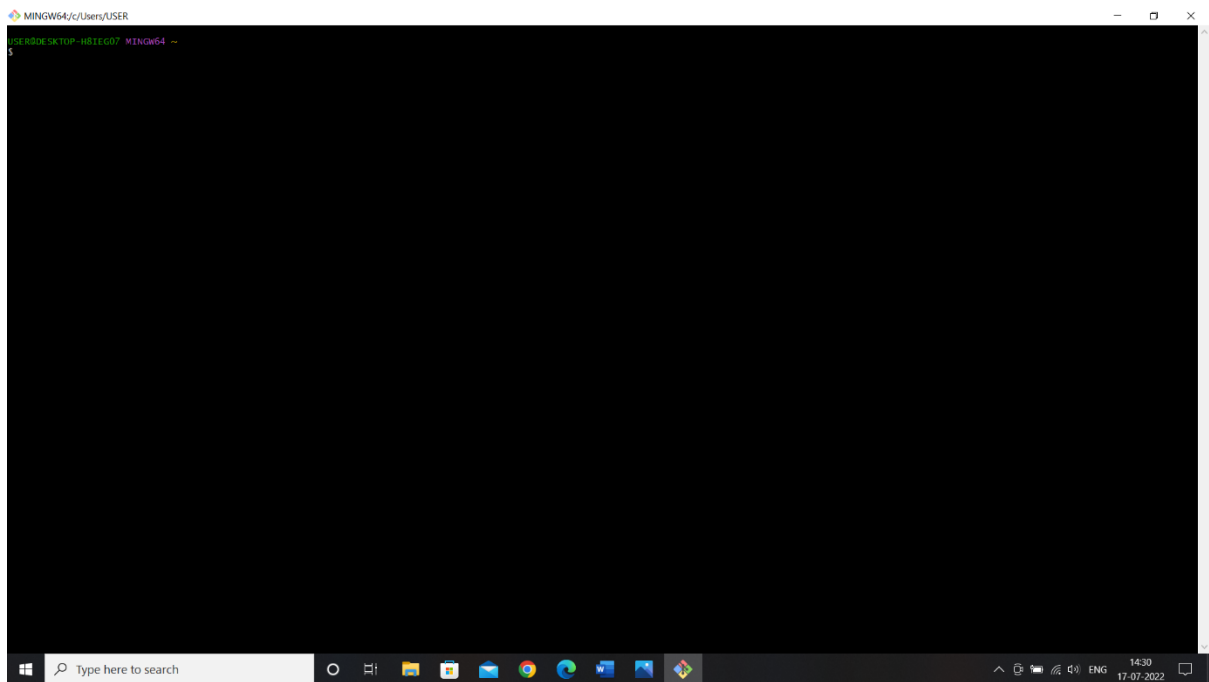
...or create a new repository on the command line

```
echo "# Demo" >> README.md
git init
git add README.md
git commit -m "First commit"
git branch -M main
git remote add origin https://github.com/vaishakbhandary/Demo.git
git push -u origin main
```

...or push an existing repository from the command line

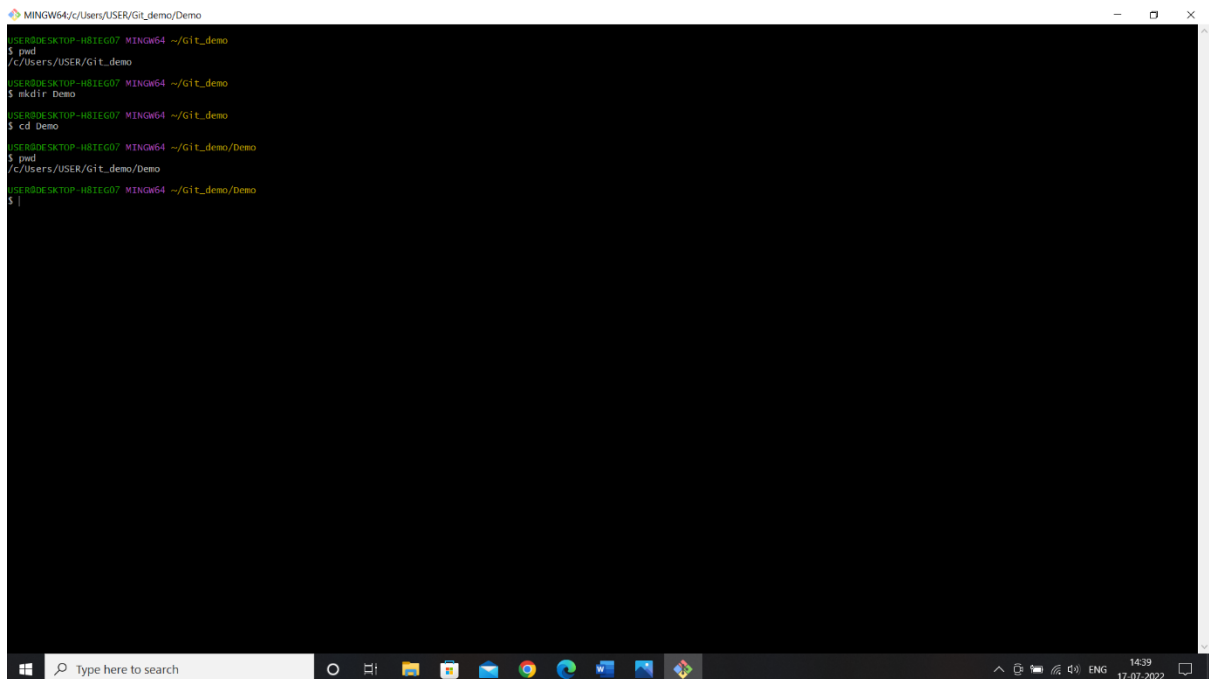
```
git remote add origin https://github.com/vaishakbhandary/Demo.git
git branch -M main
git push -u origin main
```

Step 2 :Open Git Bash.



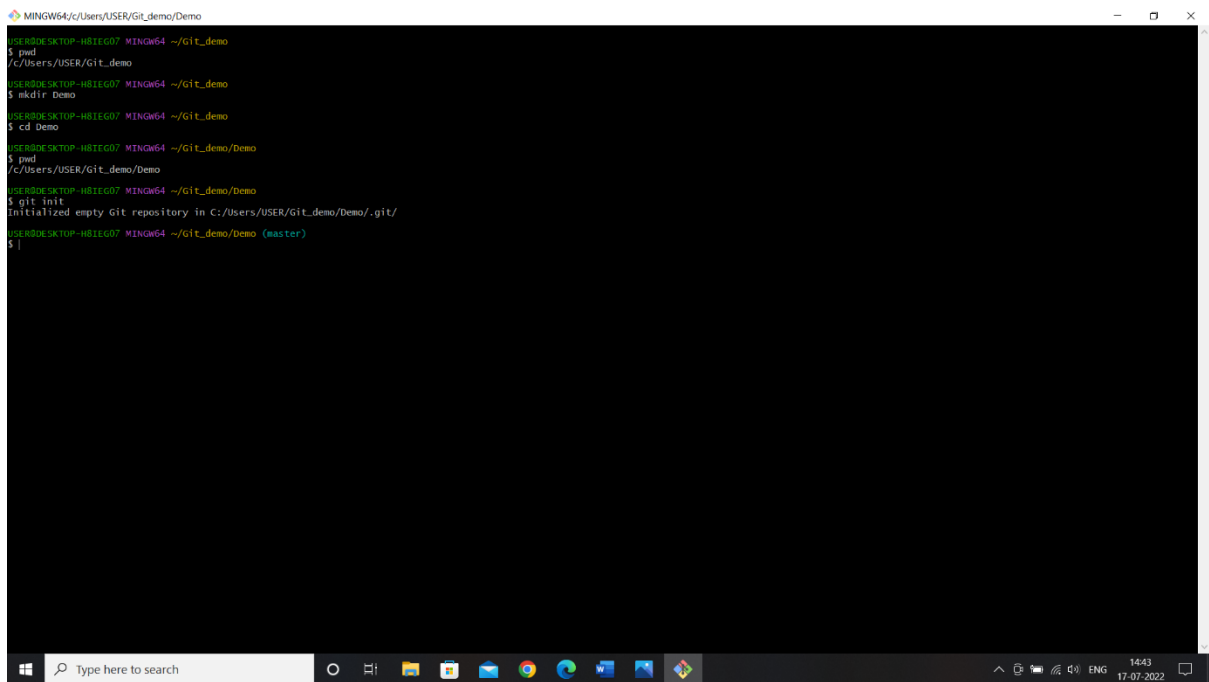
A screenshot of a Git Bash terminal window. The title bar at the top reads "MINGW64/c/Users/USER". The terminal shows the prompt "USER@DESKTOP-H8IEG07 MINGW64 ~" followed by a dollar sign "\$". The Windows taskbar is visible at the bottom, showing the search bar and several application icons. The system tray on the right indicates the time is 14:30 and the date is 17-07-2022.

Step 3 : Change the current working directory to your local project.



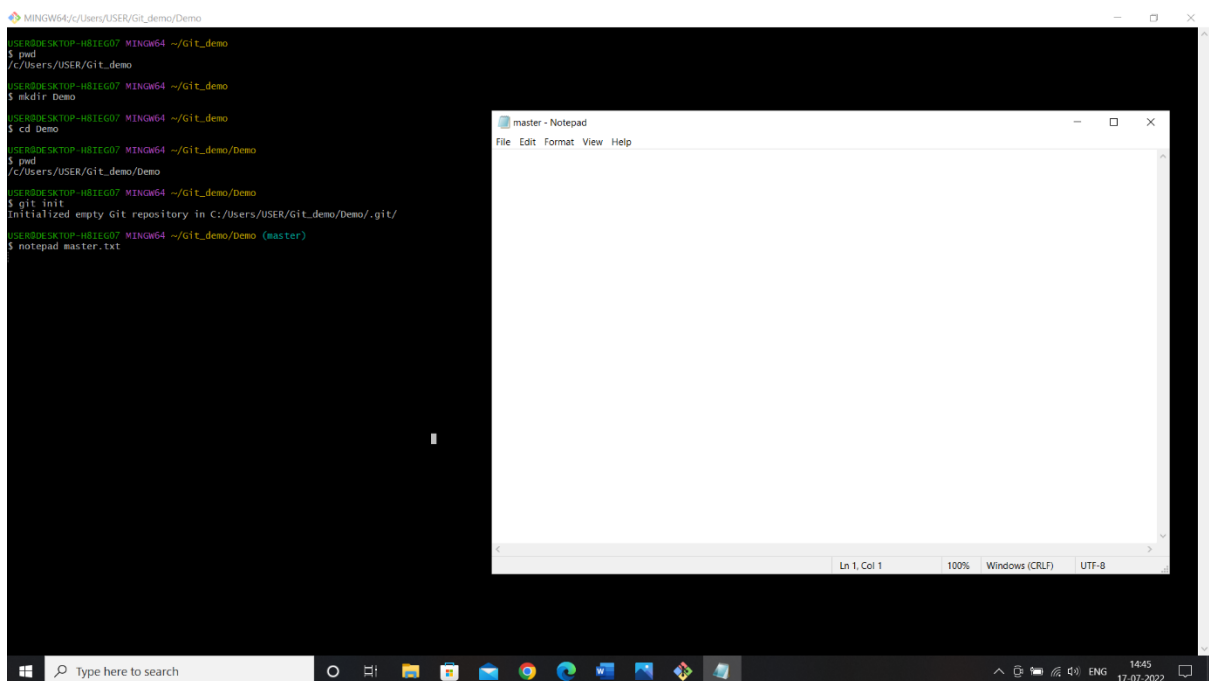
A screenshot of a Git Bash terminal window showing a series of commands to navigate to a local project directory. The title bar reads "MINGW64/c/Users/USER/git\_demo/Demo". The terminal output is as follows:  
USER@DESKTOP-H8IEG07 MINGW64 ~/git\_demo  
\$ pwd  
/c:/Users/USER/git\_demo  
USER@DESKTOP-H8IEG07 MINGW64 ~/git\_demo  
\$ mkdir Demo  
USER@DESKTOP-H8IEG07 MINGW64 ~/git\_demo  
\$ cd Demo  
USER@DESKTOP-H8IEG07 MINGW64 ~/git\_demo/Demo  
\$ pwd  
/c:/Users/USER/git\_demo/Demo  
USER@DESKTOP-H8IEG07 MINGW64 ~/git\_demo/Demo  
\$  
The Windows taskbar and system tray are also visible at the bottom, with the time now showing as 14:39.

Step 4 : Initialize the local directory as a Git repository.



```
MINGW64/c/Users/USER/Git_demo/Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo
$ pwd
/c/Users/USER/Git_demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo
$ mkdir Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo
$ cd Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo
$ pwd
/c/Users/USER/Git_demo/Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo
$ git init
Initialized empty Git repository in C:/Users/USER/Git_demo/Demo/.git/
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ |
```

Step 5: Add the files in your new local repository. This stages them for the first commit.



```
MINGW64/c/Users/USER/Git_demo/Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo
$ pwd
/c/Users/USER/Git_demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo
$ mkdir Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo
$ cd Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo
$ pwd
/c/Users/USER/Git_demo/Demo
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo
$ git init
Initialized empty Git repository in C:/Users/USER/Git_demo/Demo/.git/
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ notepad master.txt
```

master - Notepad

File Edit Format View Help

Ln 1, Col 1 100% Windows (CRLF) UTF-8

```
MINGW64/c/Users/USER/Git_demo/Demo
USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git status
On branch master

no commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git add .
USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git status
On branch master

no commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   master.txt

USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ |
```

Step 6 :Commit the files that you've staged in your local repository.

```
MINGW64/c/Users/USER/Git_demo/Demo
USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git status
On branch master

no commits yet

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

nothing added to commit but untracked files present (use "git add" to track)
USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git add .
USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git status
On branch master

no commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   master.txt

USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git commit -m "first commit"
[master (root-commit) 9b48e5f] First commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 master.txt
USERDESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ |
```



vaishakbhandary / Demo (Public)

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

### Quick setup — if you've done this kind of thing before

or

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

#### ...or create a new repository on the command line

```

echo "# Demo" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/vaishakbhandary/Demo.git
git push -u origin main
    
```

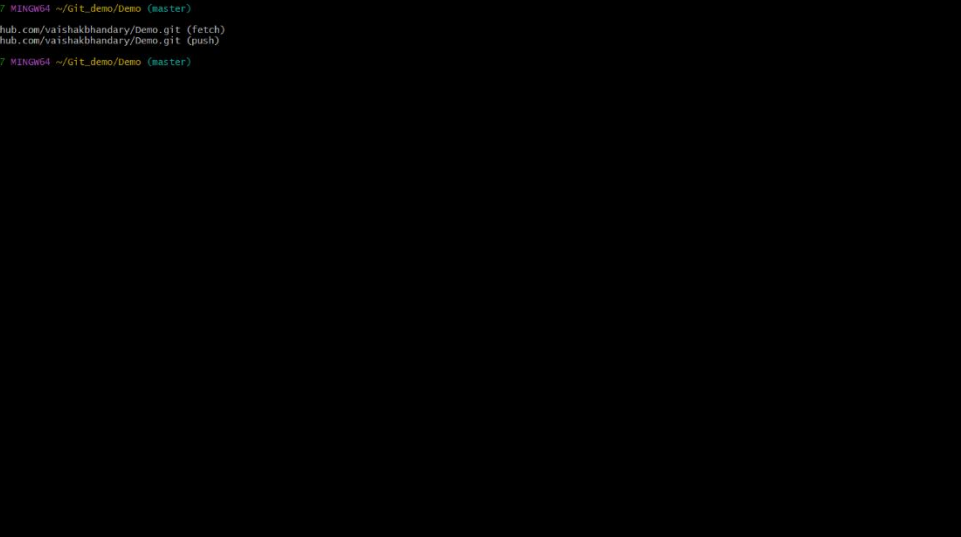
#### ...or push an existing repository from the command line

```

git remote add origin https://github.com/vaishakbhandary/Demo.git
git branch -M main
git push -u origin main
    
```

#### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.



The screenshot shows a Windows terminal window with the title bar "MINGW64/c:/Users/USER/Git\_demo/Demo". The terminal content is as follows:

```
USER@DESKTOP-HBTEG07 MINGW64 ~/Git_demo/Demo (master)
$ git remote add origin https://github.com/vaishakbhbandary/Demo.git

USER@DESKTOP-HBTEG07 MINGW64 ~/Git_demo/Demo (master)
$ git remote -v
origin https://github.com/vaishakbhbandary/Demo.git (fetch)
origin https://github.com/vaishakbhbandary/Demo.git (push)

USER@DESKTOP-HBTEG07 MINGW64 ~/Git_demo/Demo (master)
$
```

The taskbar at the bottom shows the Windows Start button, a search bar with the text "Type here to search", and several application icons including File Explorer, Mail, Chrome, and others. The system tray on the right indicates the time as 14:52 on 17-07-2022.

## Step 9 : Push the changes in your local repository to GitHub.com.

The screenshot illustrates the process of pushing local changes to a GitHub repository. The top portion shows a terminal window with the following commands and output:

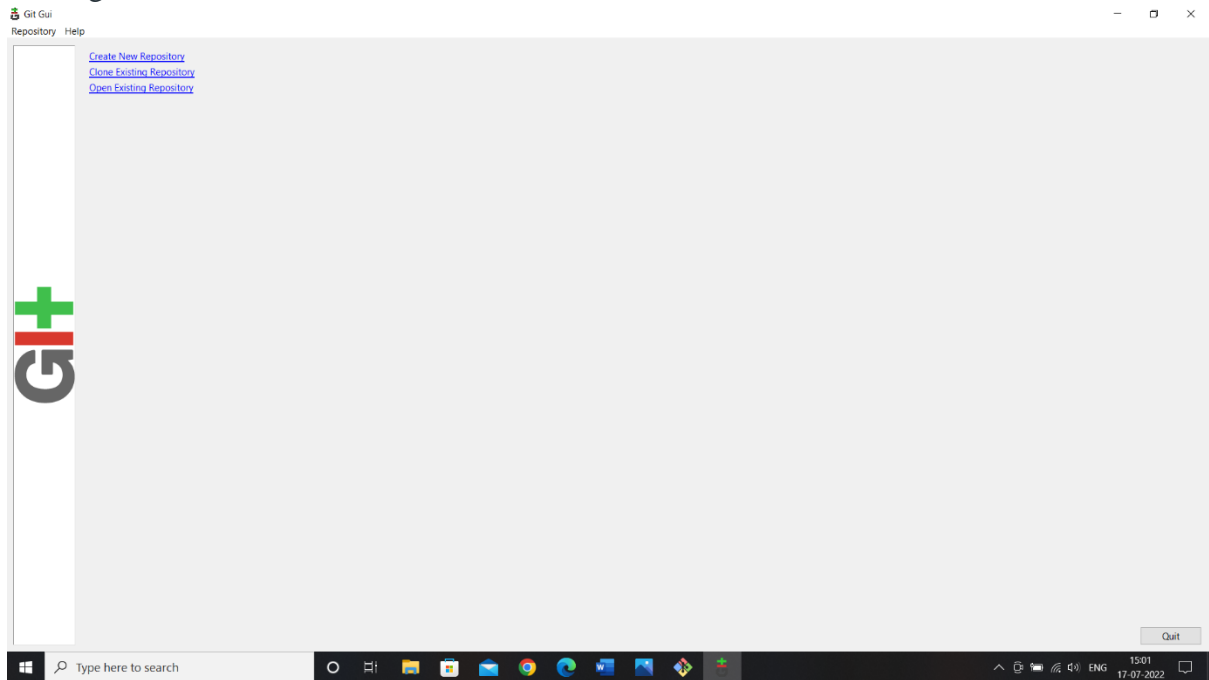
```
USER@DESKTOP-H8IEG07 MINGW64 ~/Git_demo/Demo (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 214 bytes | 107.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/vaishakbhandary/Demo.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

The bottom portion shows the GitHub repository page for `vaishakbhandary/Demo`. The page displays the repository's status, including the `master` branch, 1 commit, and 1 file (`master.txt`). It also includes a prompt to add a README file and sections for Releases and Packages.

## GUI:

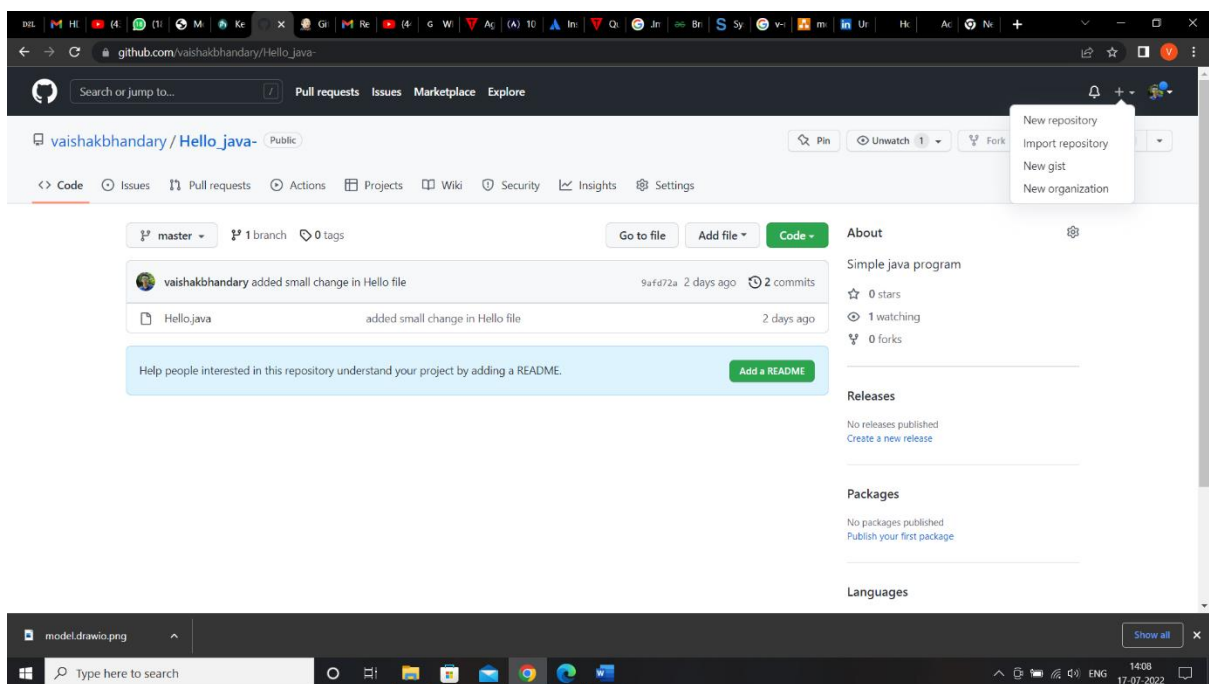
Step 1 : Go to Start > All Programs > Git > Git GUI and make a Desktop Shortcut.

### Getting started with Git GUI

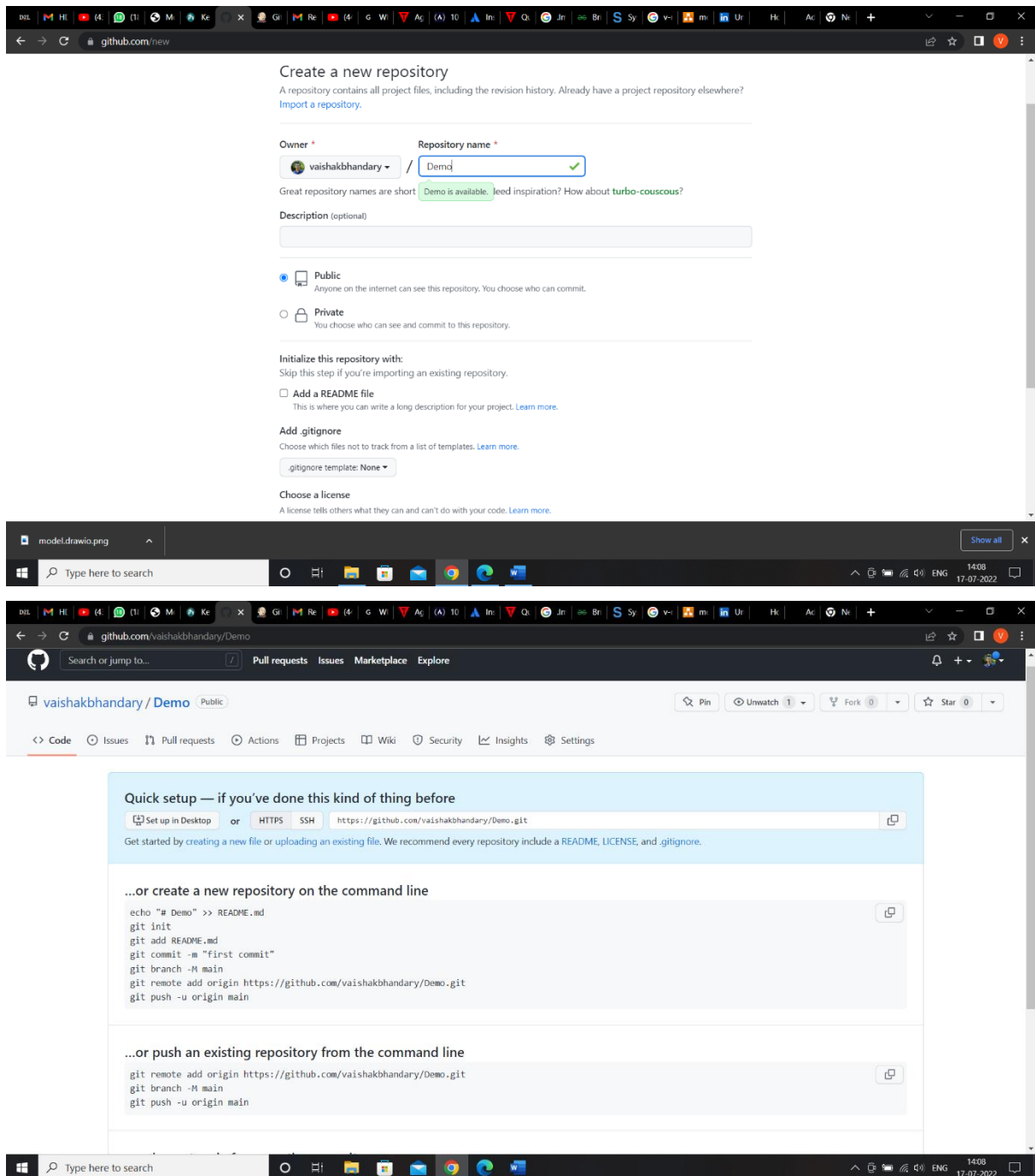


### Step 2 : Create Remote Repository

Now, we need a Git repository



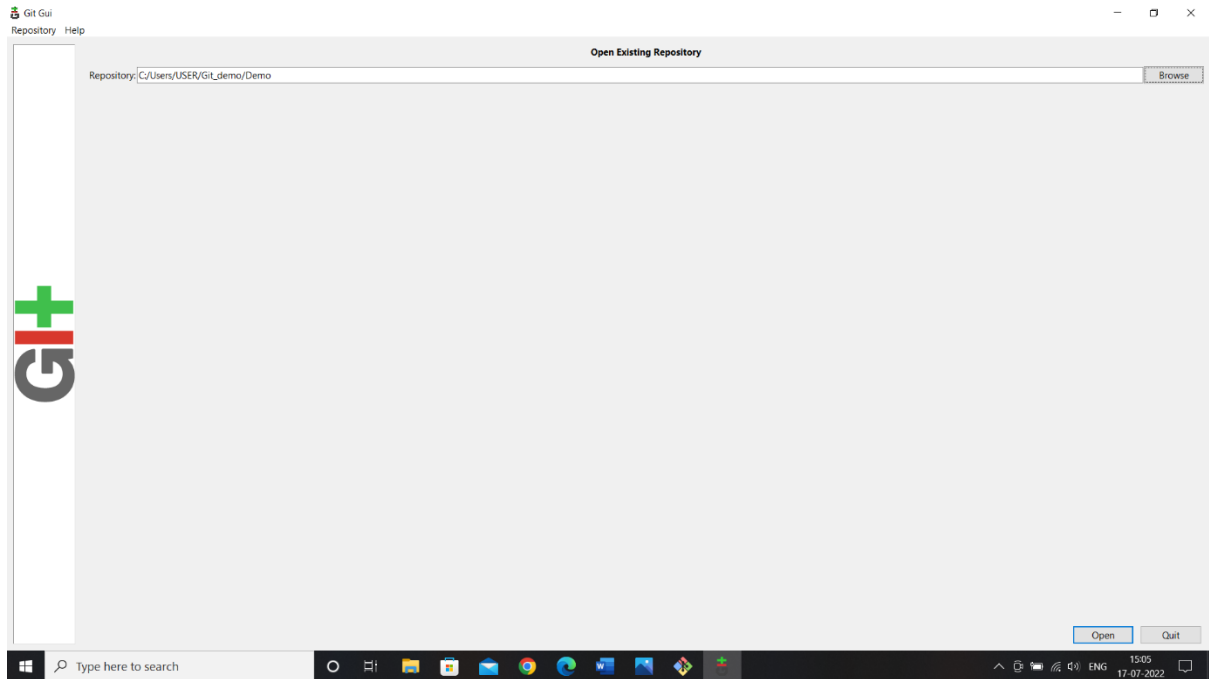
ry, and we'll create a new remote repository on Github.



### Step 3 : Create a Local Repository

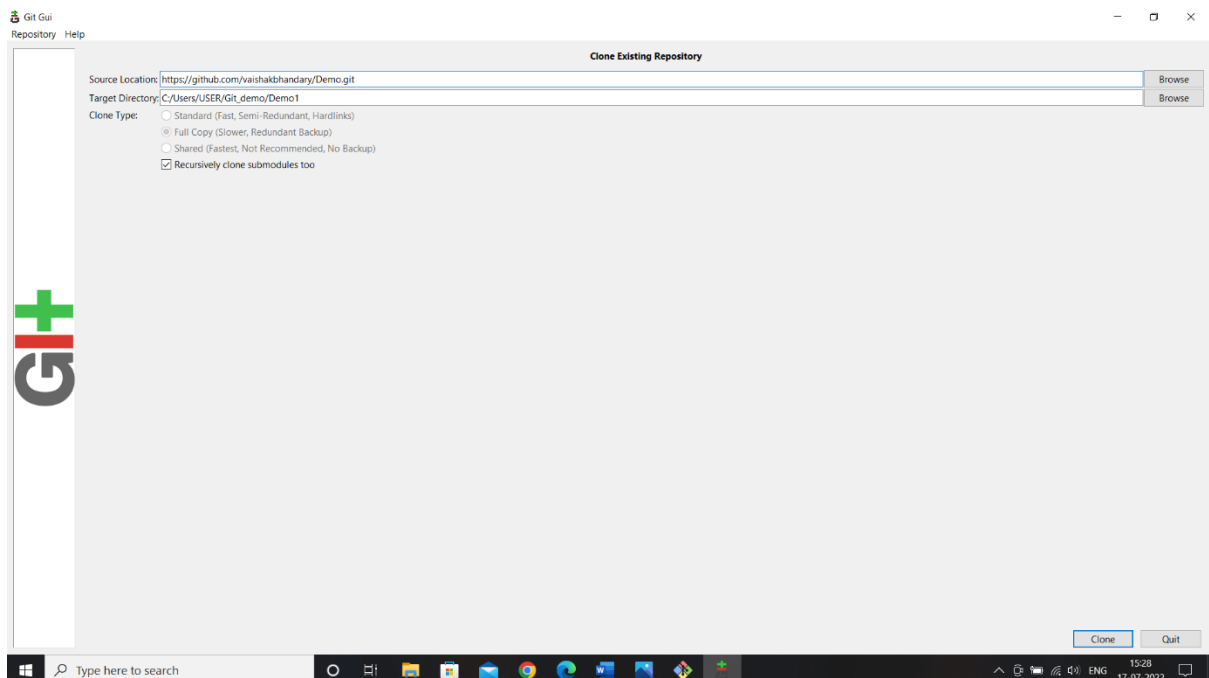
For creating a local repository: in our Git GUI, click on “**Create New Repository**”.

Select the location you wish to store your repository in. It is important to note that the selected repository location **MUST NOT** exist.



#### Step 4 : Clone a Remote Repository to a Local Repository

In order to clone a repository, click on the “**Clone Existing Repository**” link in the Git GUI window. An existing repository is one that is already initialized and/or has commits pushed to it.

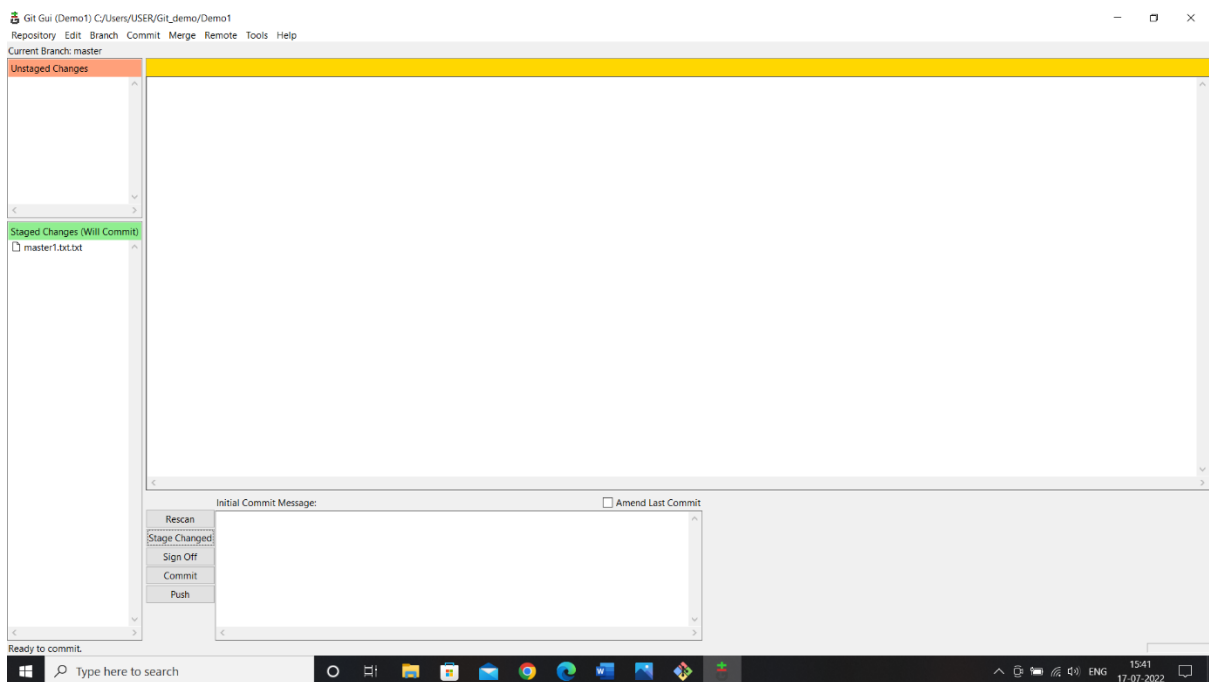
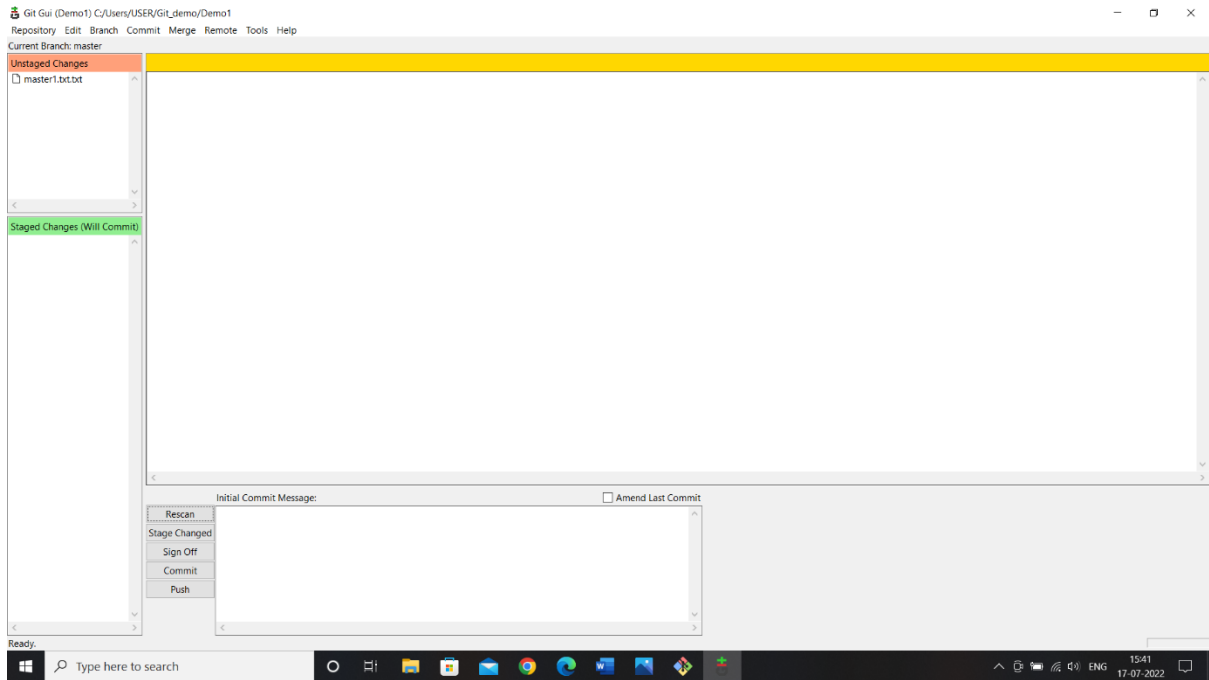


## Step 5 : Working with the GUI Client

The Git GUI makes it easier to perform Git-related tasks, such as staging changes, commits, and pushes.

### Staged Changes

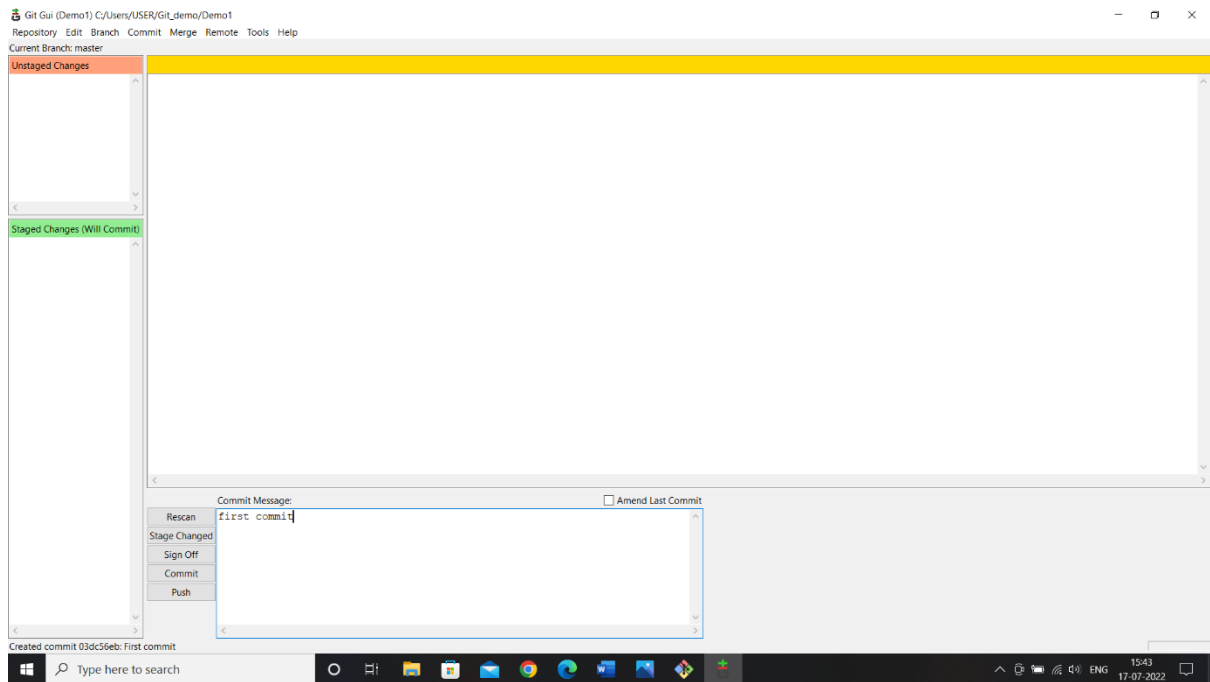
When we move files to a Git directory, you will see all the files in the “Unstaged Changes” window. This basically means that new files have been added, removed, updated, etc. When we click the “Stage Changed” button, it will attempt to add all the new files to the Git index.



Step 6 :

### Commits :

After we've staged your changes, we need to commit them to your local repository. Type a Commit Message that makes sense to the changes that were made. When we are done, press the Commit button.



Step 7 :

Pushing:

After we have committed all the codes in the local repository, we need to push these changes to our remote repository on GitHub. Without pushing the changes, others would not be able to access the code.

