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.

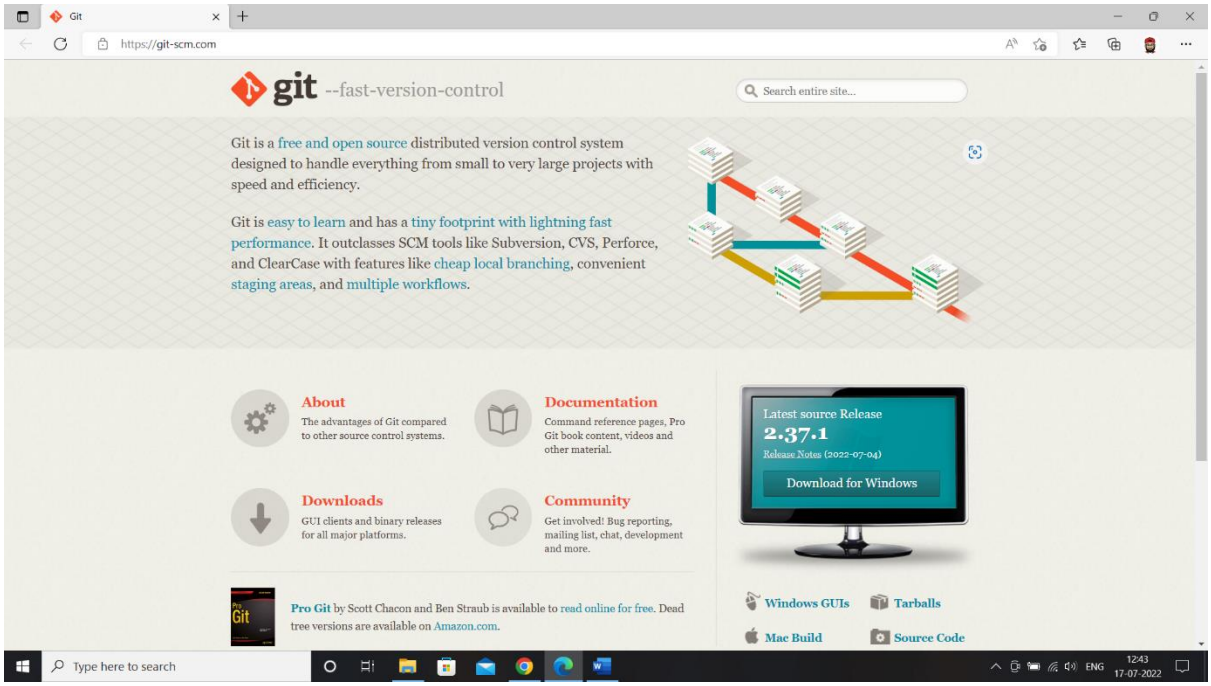
Git bash :

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 :

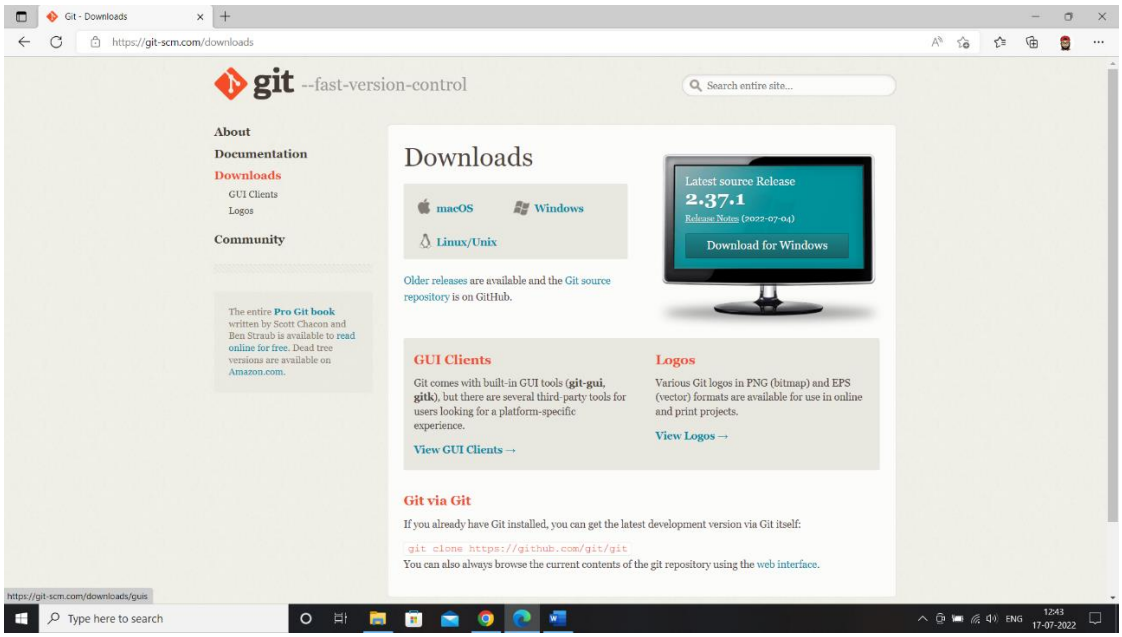
Step 1 :

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



Step 2

Download the installer.



Step 3

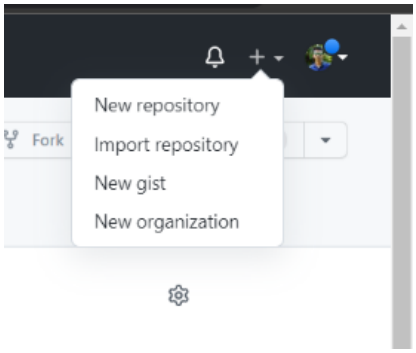
Run the .exe file you just downloaded and follow the instructions in the installer.

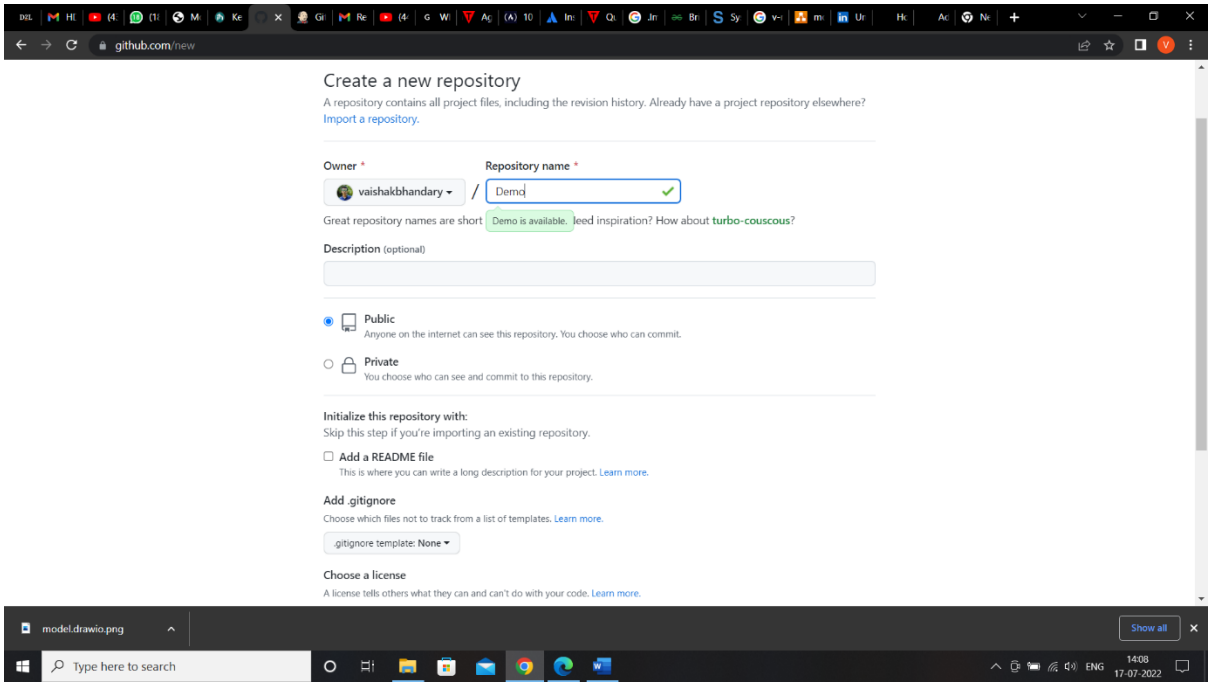
Step 4

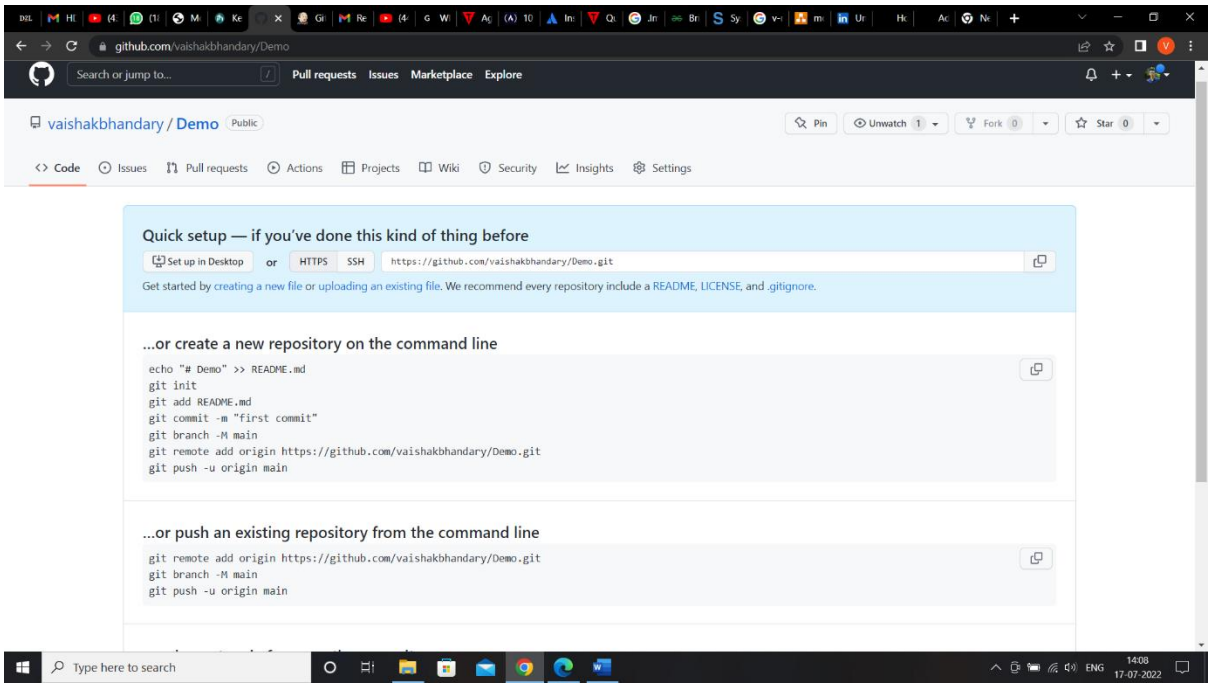
Run Git Bash by right-clicking on any folder and selecting the **Git Bash Here** option from the context menu(right-click menu).

<u>Git Terminal :</u>

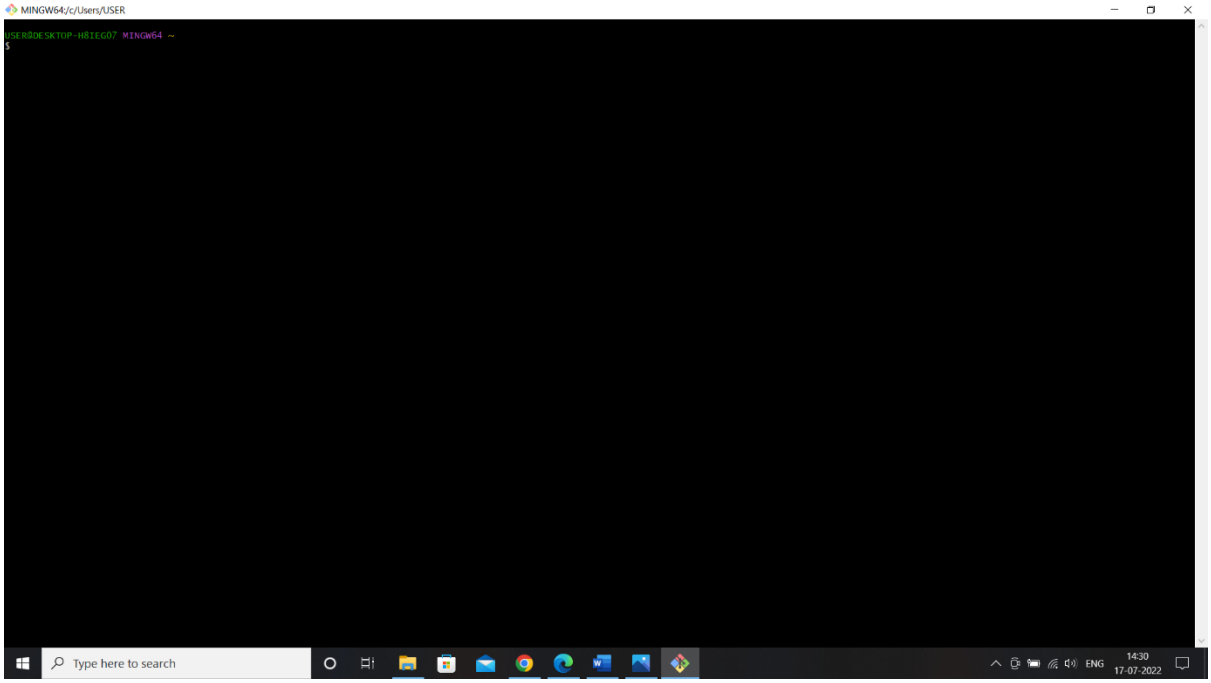
Step 1 : Create a new repository on GitHub.com. To avoid errors, do not initialize the new repository with README, license, or gitignore files. You can add these files after your project has been pushed to GitHub.



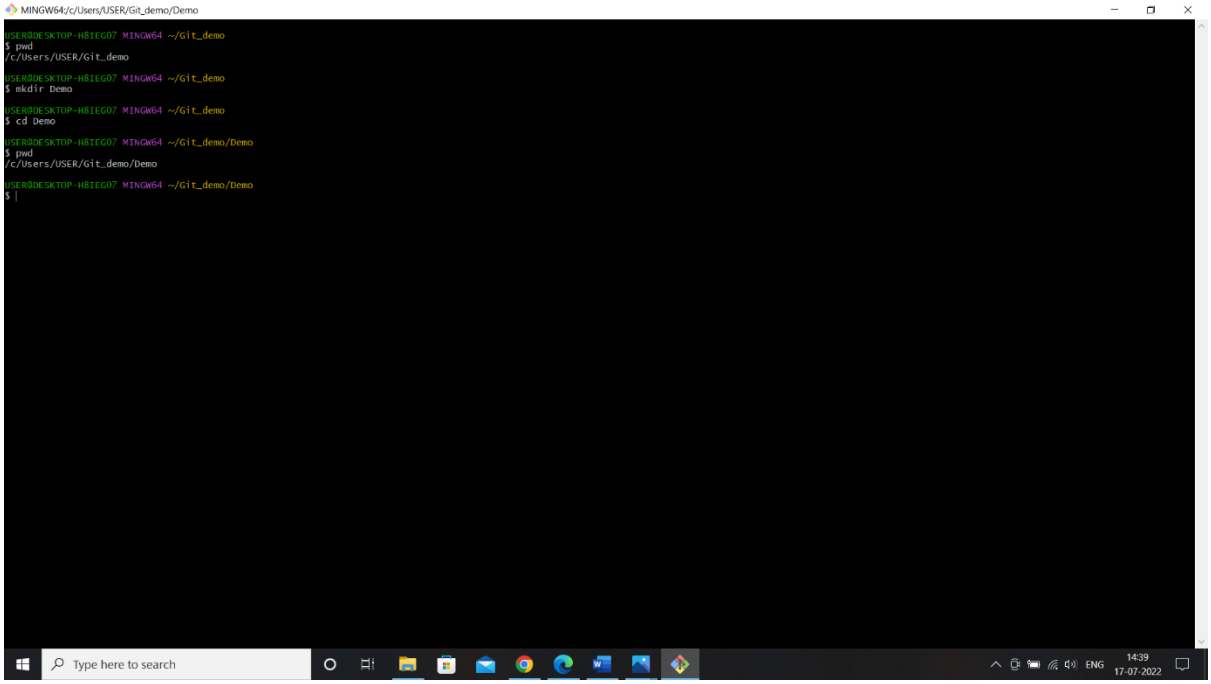

github.com/new
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Owner *
Repository name *
vaishakbhandary
Demo
Great repository names are short
Demo is available.
Need inspiration? How about 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
ENG
14:08
17-07-2022


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
https://github.com/vaishakbhandary/Demo.git
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
Type here to search
ENG
14:08
17-07-2022

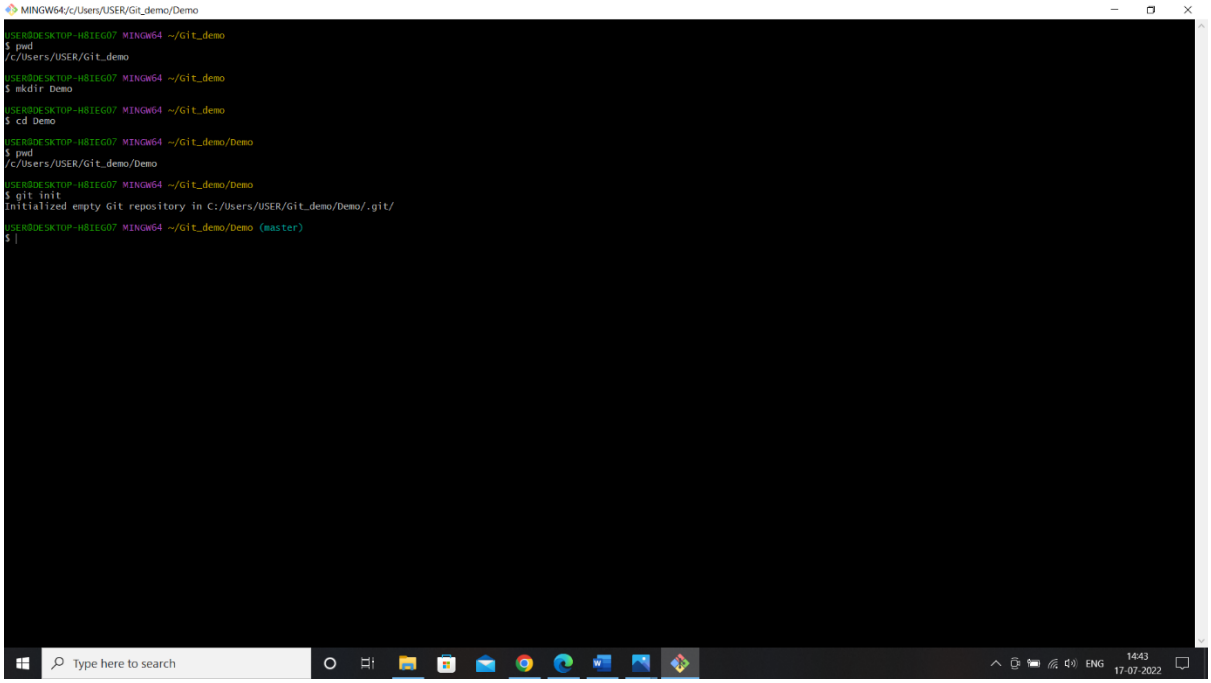
Step 2 :Open Git Bash.



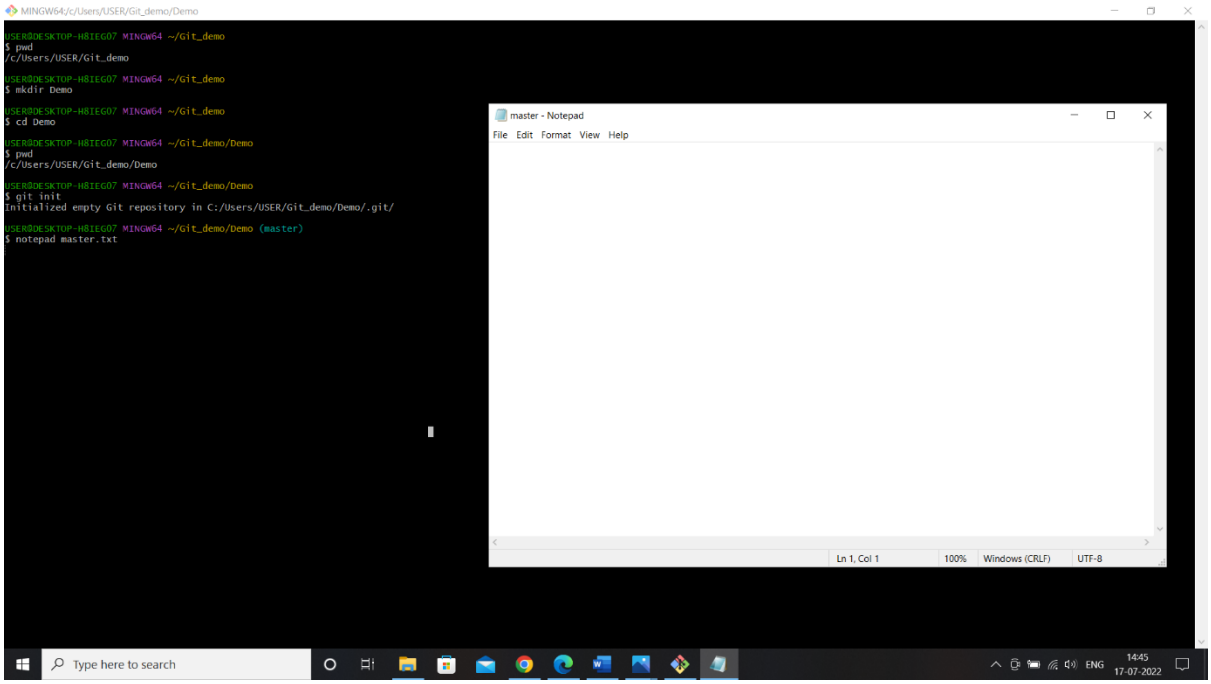
Step 3 : Change the current working directory to your local project.

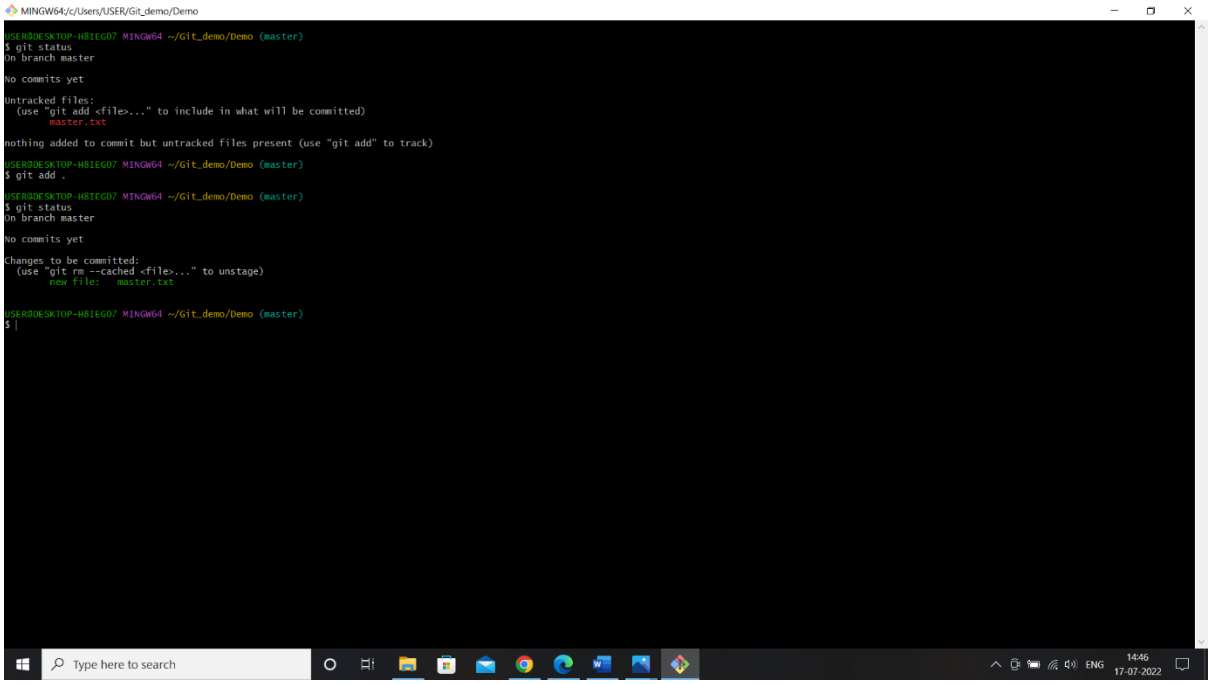


Step 4 : Initialize the local directory as a Git repository.

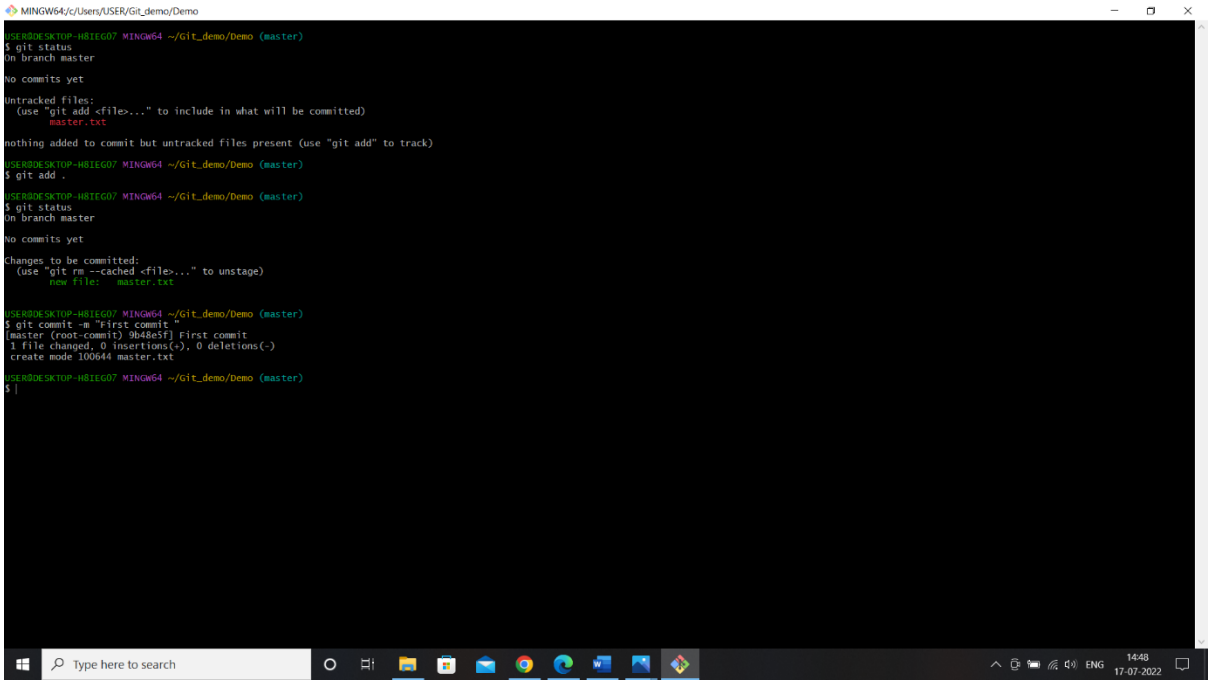


Step 5: Add the files in your new local repository. This stages them for the first commit.

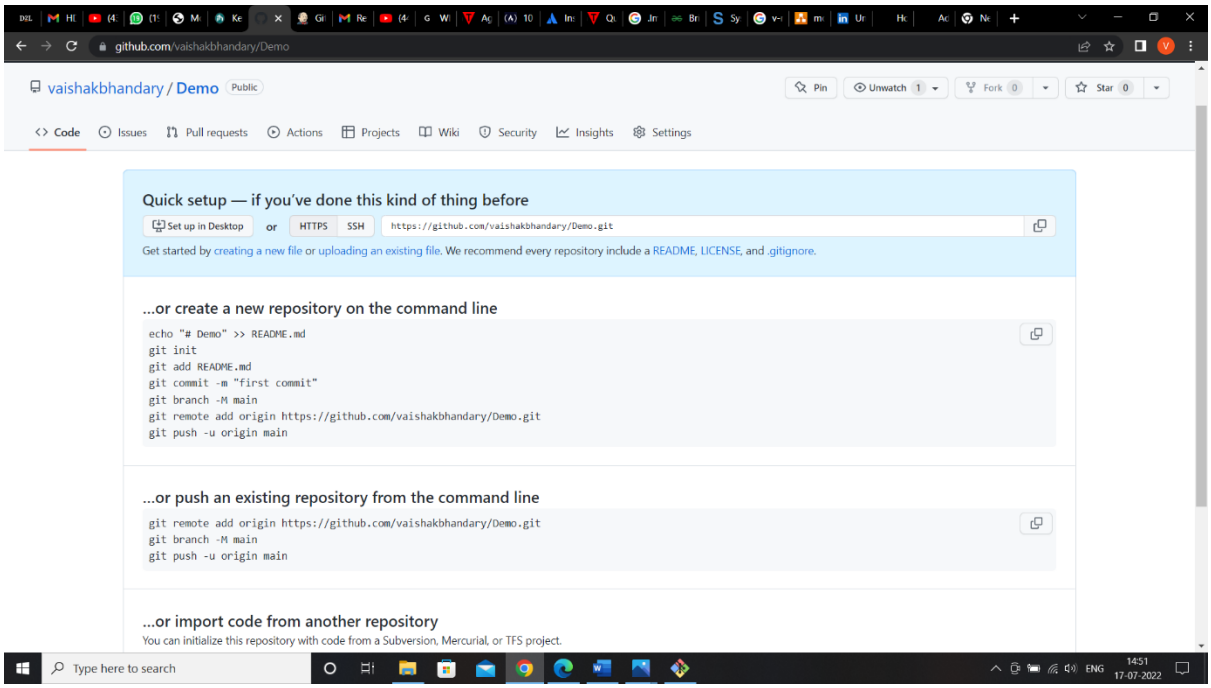




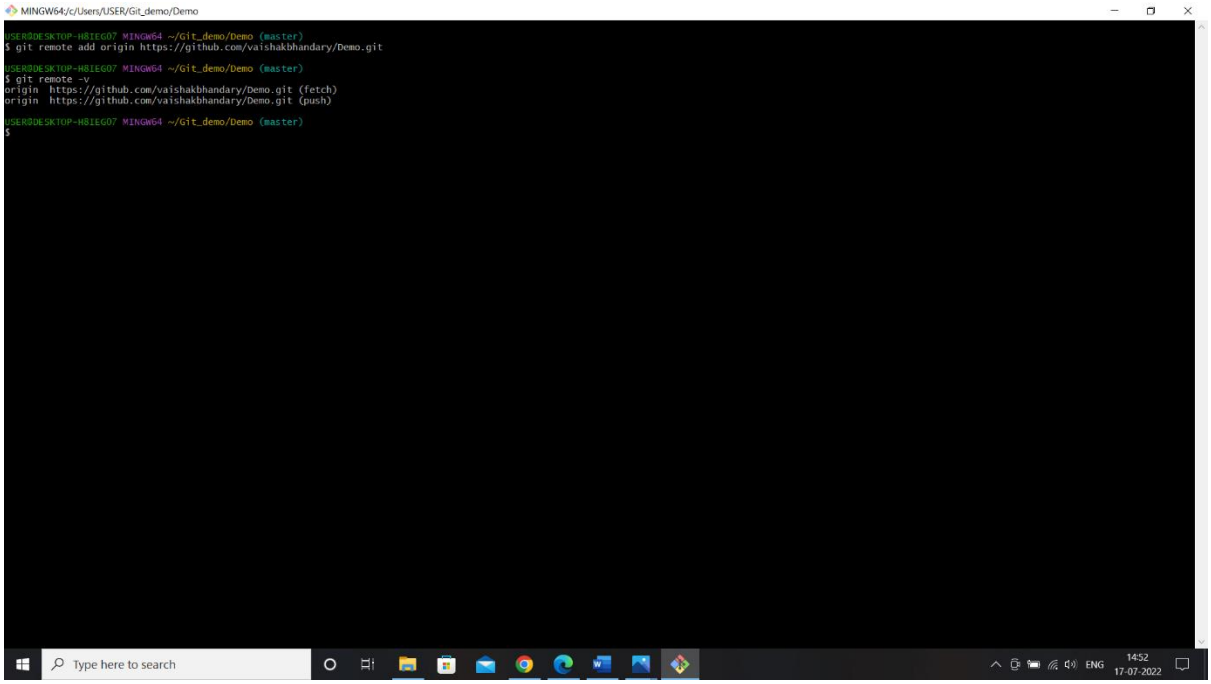
Step 6 :Commit the files that you've staged in your local repository.



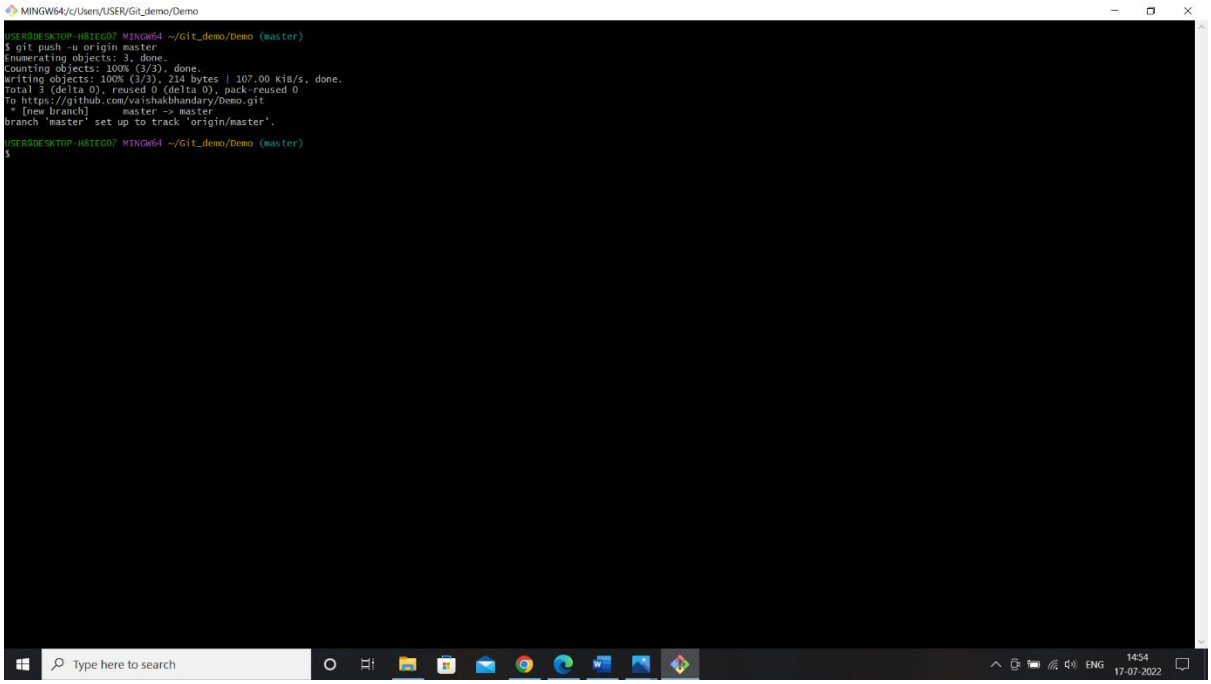
Step 7 : At the top of your repository on GitHub.com's Quick Setup page, click to copy the remote repository URL.

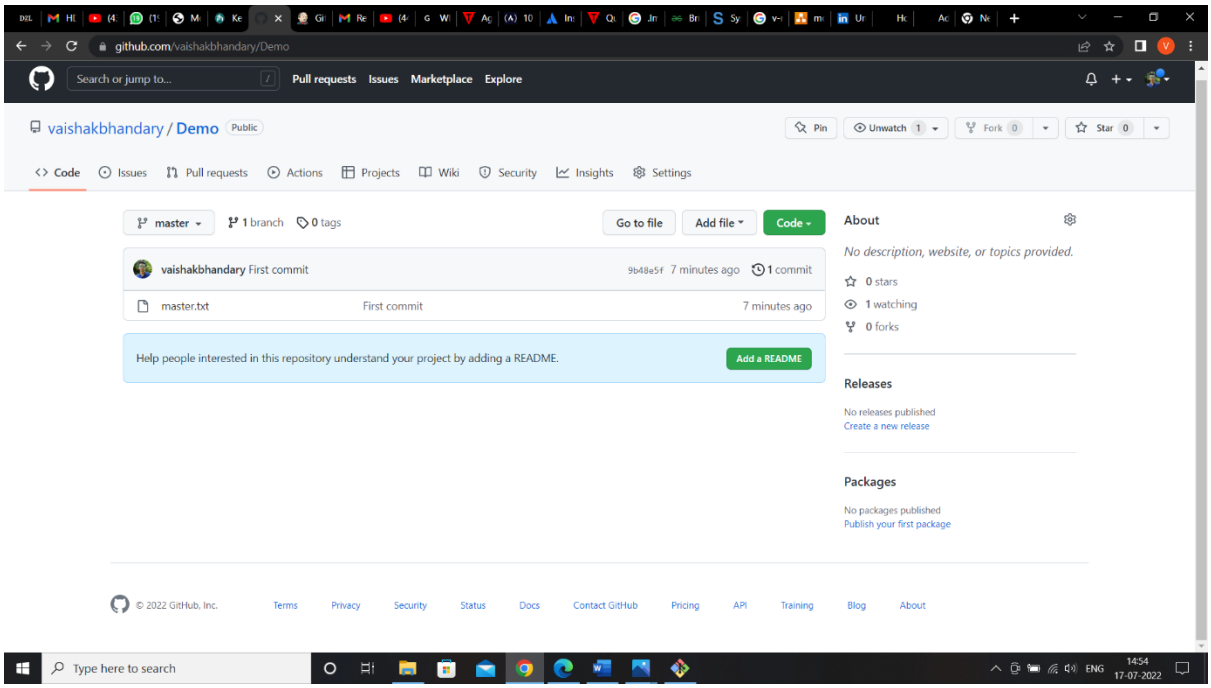


Step 8 : In the Command prompt, add the URL for the remote repository where your local repository will be pushed.



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

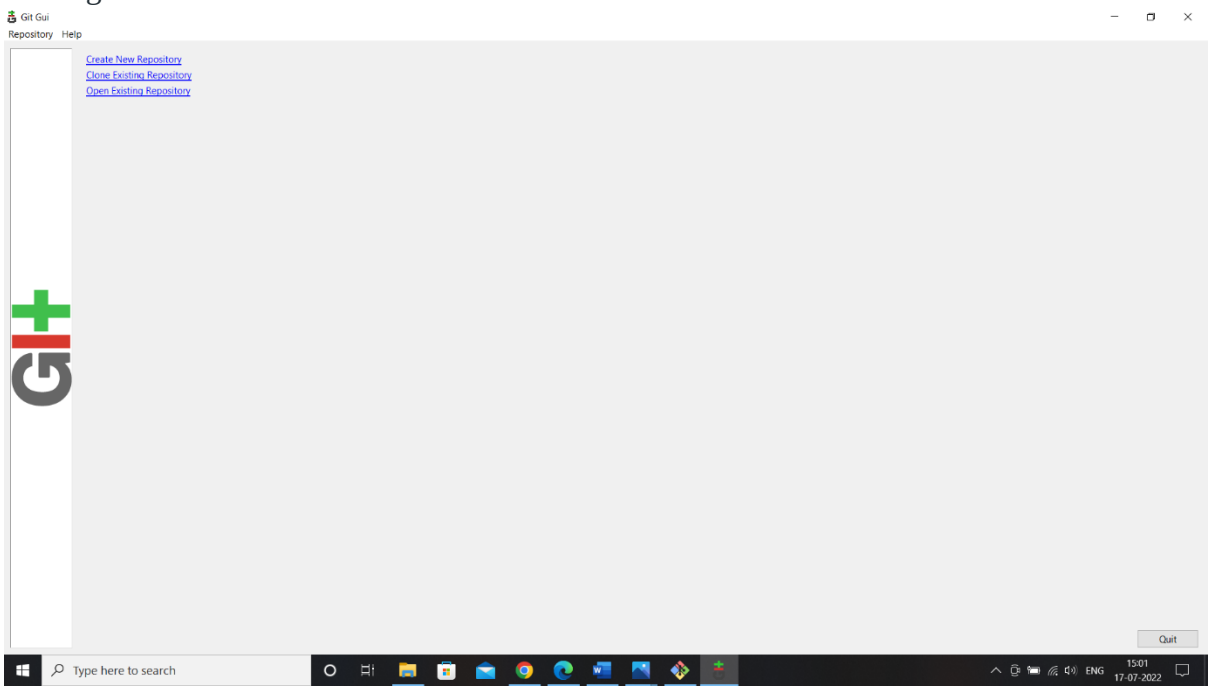




GUI :

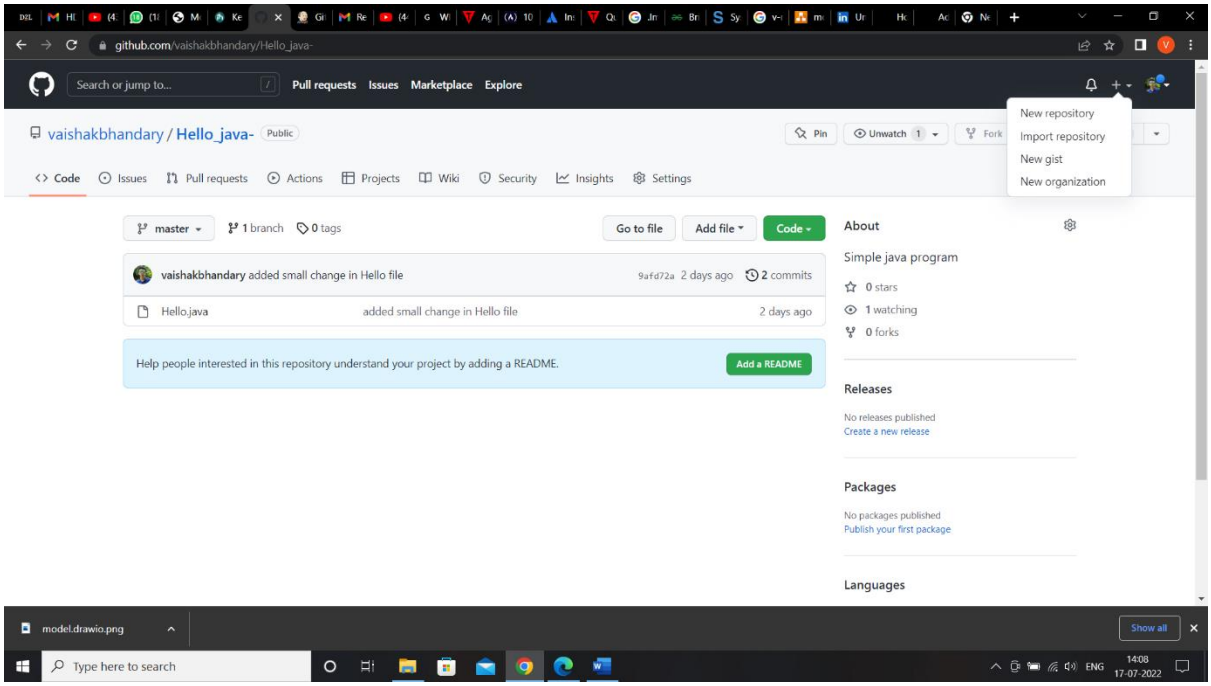
Step 1 : Go to Start > All Programs > Git > Git GUI and make a Desktop Shortcut.

Getting started with Git GUI

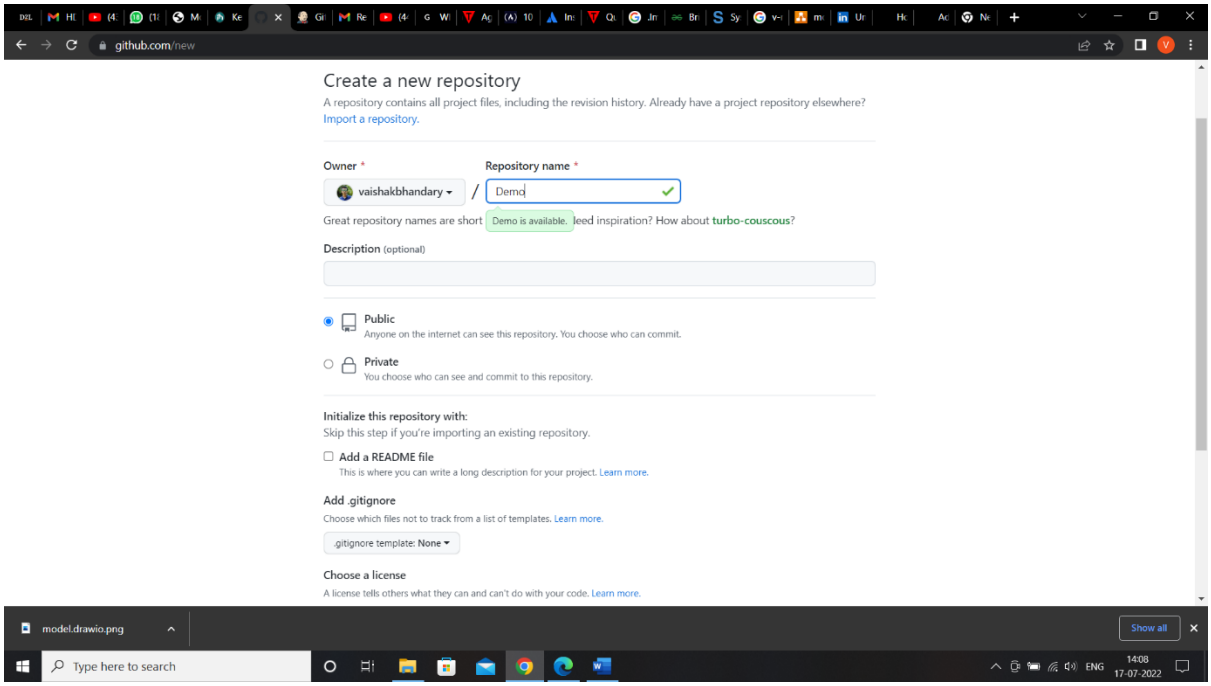


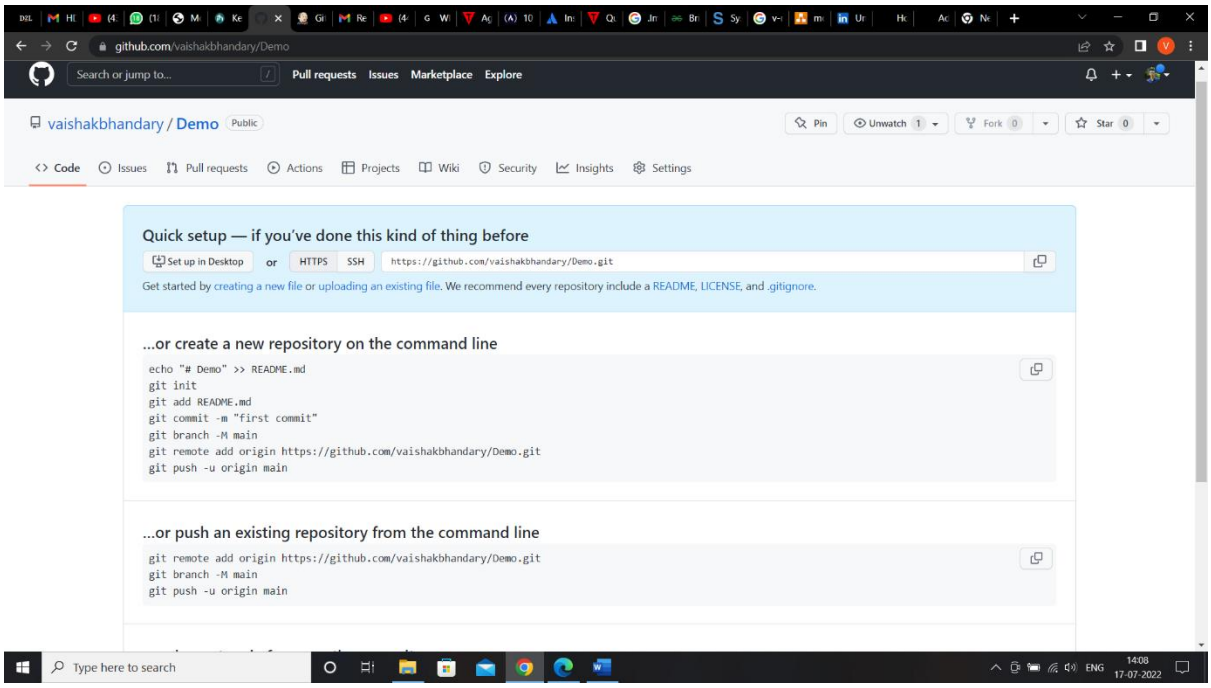
Step 2 : **Create Remote Repository**

Now, we need a Git repository



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

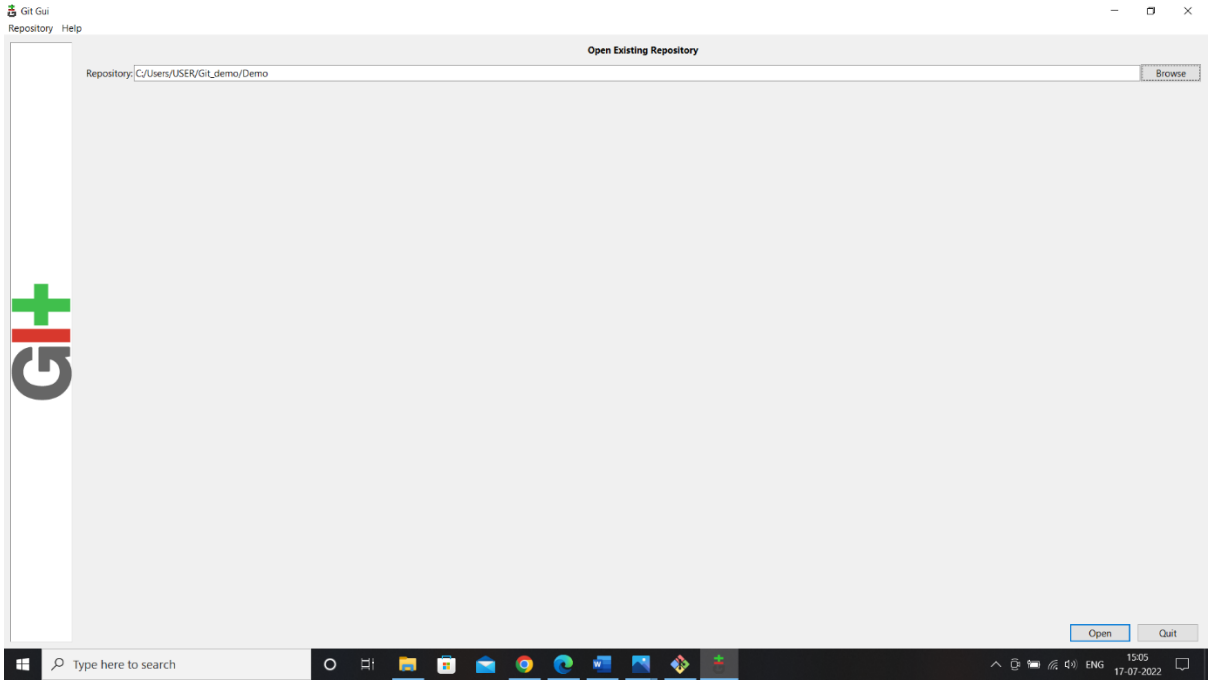




Step 3 : **Create a Local Repository**

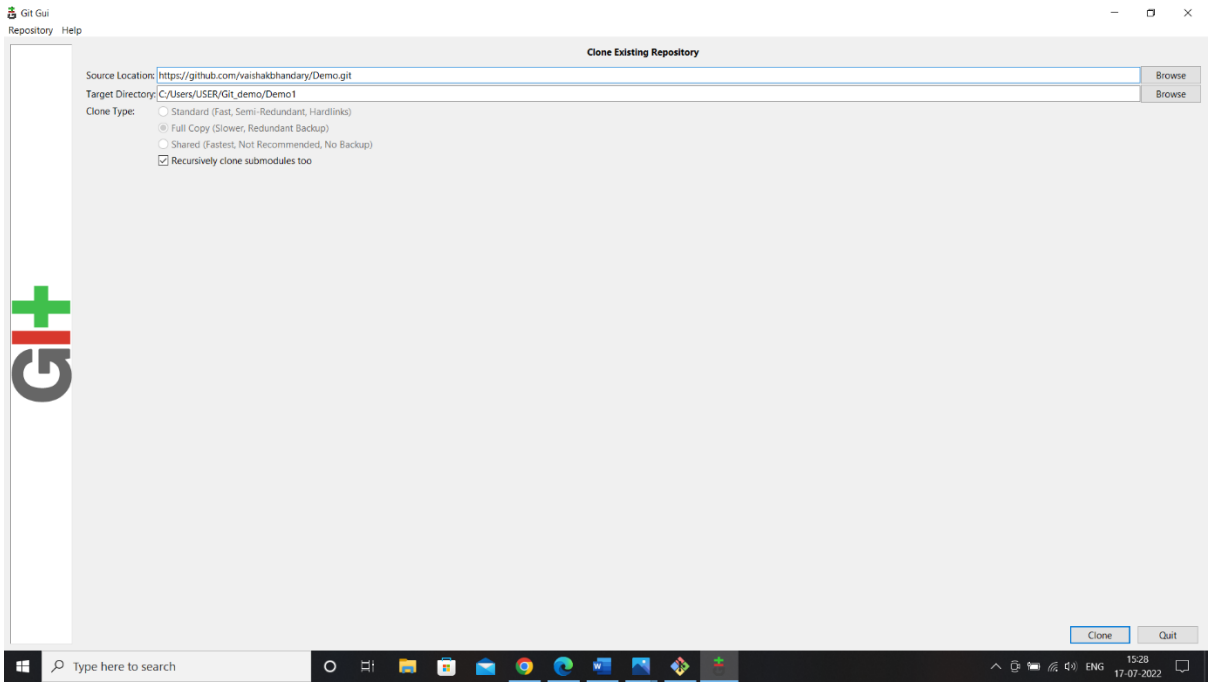
For creating a local repository: in our Git GUI, click on **"Create New Repository"**. Select the location you wish to store your repository in. It is important to note that the selected repository location

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.

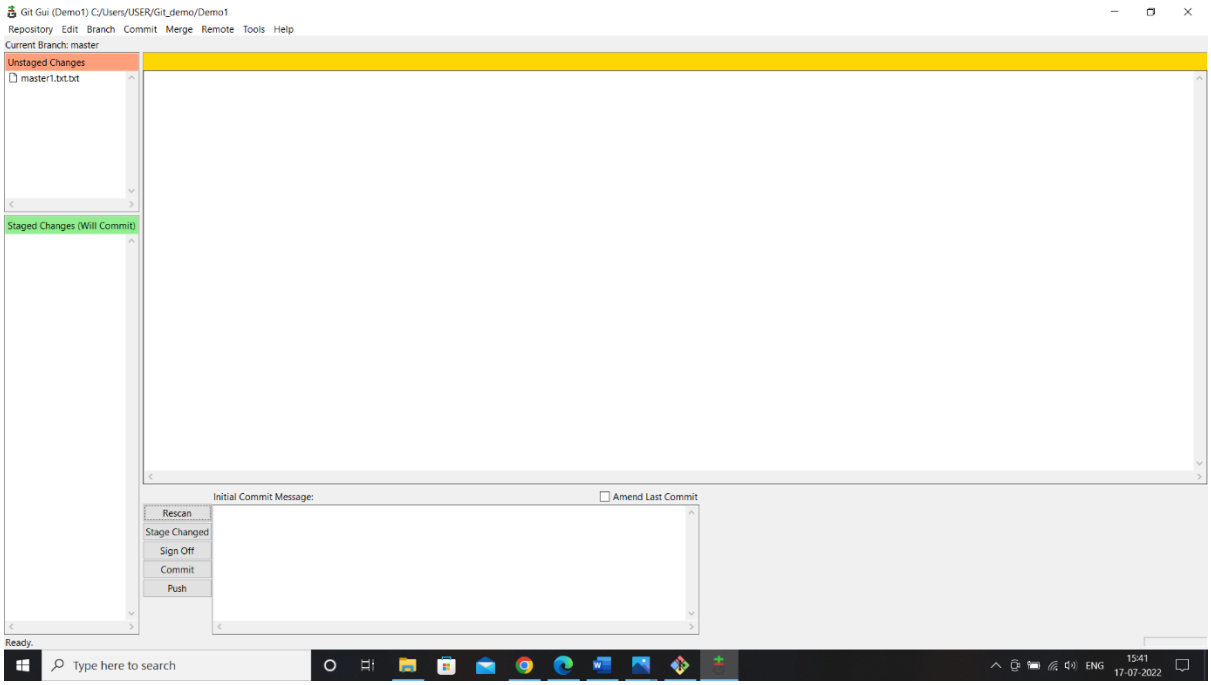


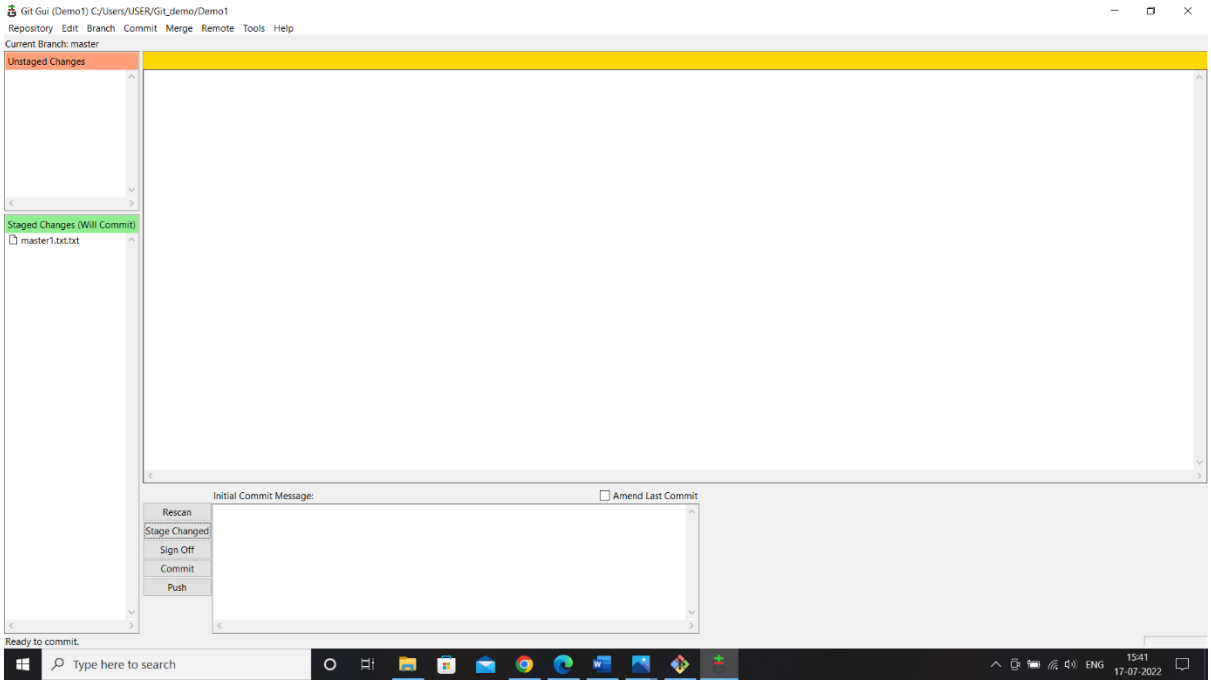
Step 5 : Working with the GUI Client

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.

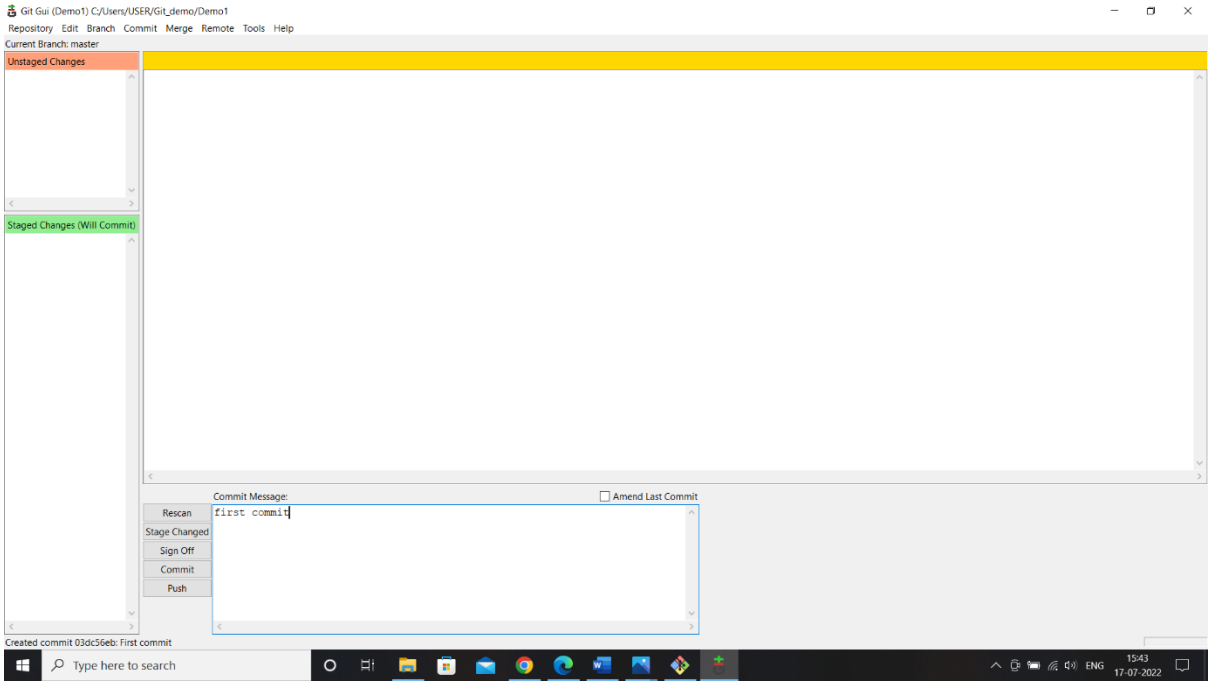




Step 6 :

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.