**Working on Git Bash**

* Difficulty Level : [Basic](https://www.geeksforgeeks.org/basic/)
* Last Updated : 04 Aug, 2021

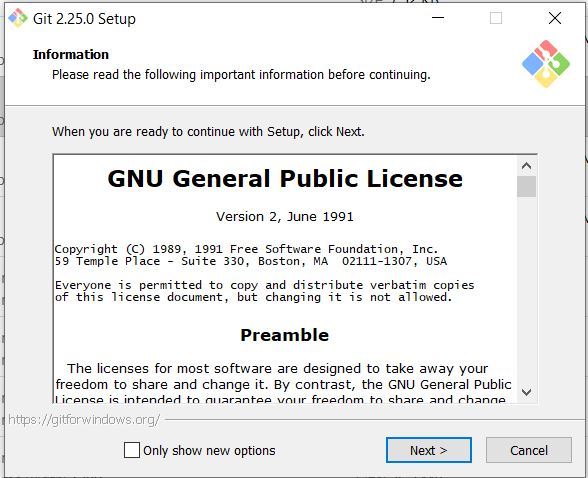
Git Bash is an application that provides Git command line experience on the Operating System. It is a command-line shell for enabling git with the command line in the system. A shell is a terminal application used to interface with an operating system through written commands. Git Bash is a package that installs Bash, some common bash utilities, and Git on a Windows operating system. In Git Bash the user interacts with the repository and git elements through the commands.

**What is Git?**

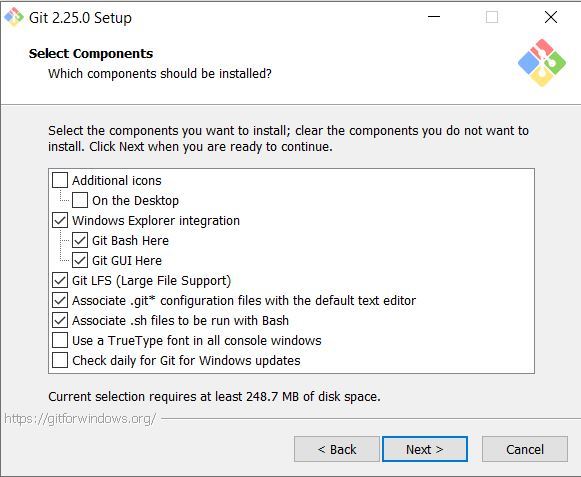
* [Git](https://www.geeksforgeeks.org/git-lets-get-into-it/) is version-control system for tracking changes in source code during software development.
* It is designed for coordinating work among programmers, but it can be used to track changes in any set of files.
* Its goal is to increase efficiency, speed and easily manage large projects through version controlling.
* Every git working directory is a full-fledged repository with complete history and full version-tracking capabilities, independent of network access or a central server.
* Git helps the team cope up with the confusion that tends to happen when multiple people are editing the same files.

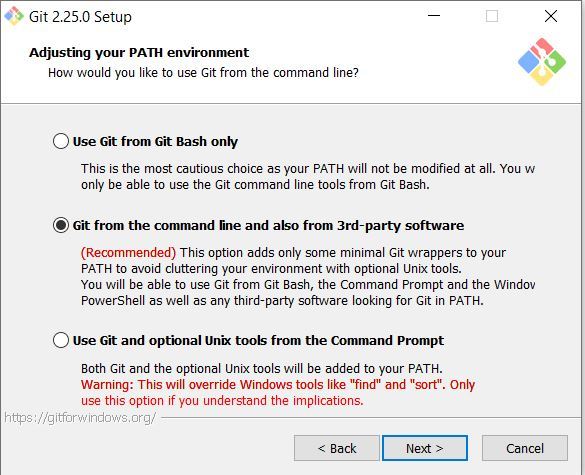
**Installing Git Bash**

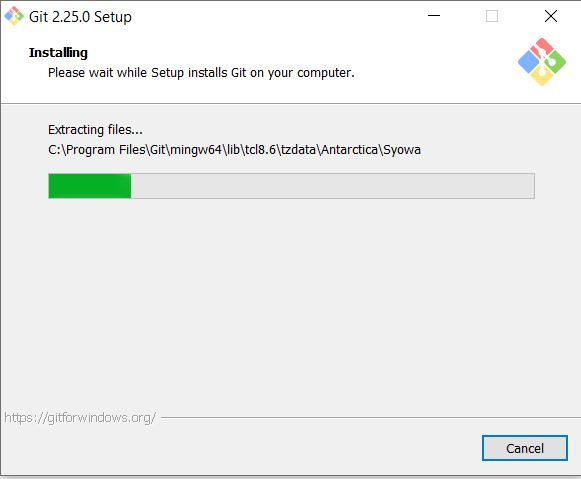
Follow the steps given below to install Git Bash on Windows:

**Step 1:** The .exe file installer for Git Bash can be downloaded from <https://gitforwindows.org/>  
Once downloaded execute that installer, following window will occur:-  


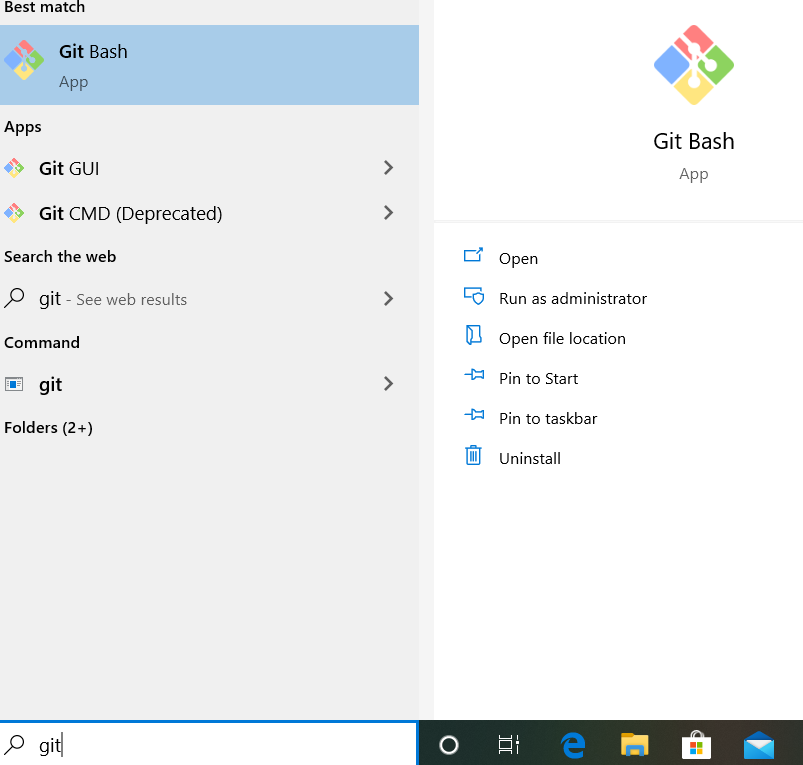
**Step 2:** Select the components that you need to install and click on the Next button.



**Step 3:** Select how to use the Git from command-line and click on Next to begin the installation process.  


**Step 4:** Let the installation process finish to begin using Git Bash.  


To open Git Bash navigate to the folder where you have installed the git otherwise just simply search in your OS for git bash.



**Navigate in Git Bash**

**cd command**

cd command refers to **change directory** and is used to get into the desired directory.

To navigate between the folders the **cd** command is used  
**Syntax:**

cd folder\_name

**ls command**

ls command is used to list all the files and folders in the current directory.  
**Syntax:**

ls

**Set your global username/email configuration**

Open Git Bash and begin creating a username and email for working on Git Bash.

**Set your username:**

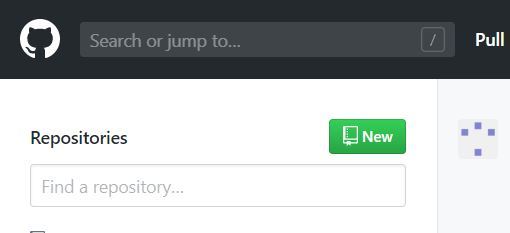
git config --global user.name "FIRST\_NAME LAST\_NAME"

**Set your email address:**

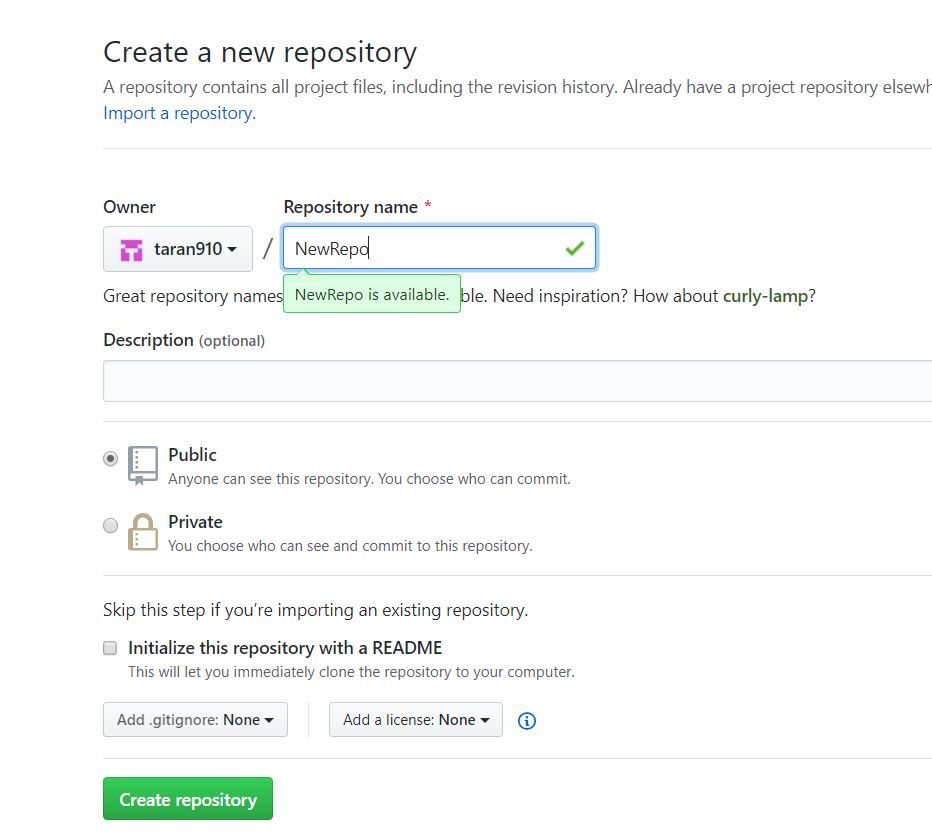
git config --global user.email "MY\_NAME@example.com"

**Initializing a Local repository**

Follow the steps given below to initialize your Local Repository with Git:

**Step 1:** Make a repository on [Github](https://www.geeksforgeeks.org/working-on-git-bash/www.github.com)  


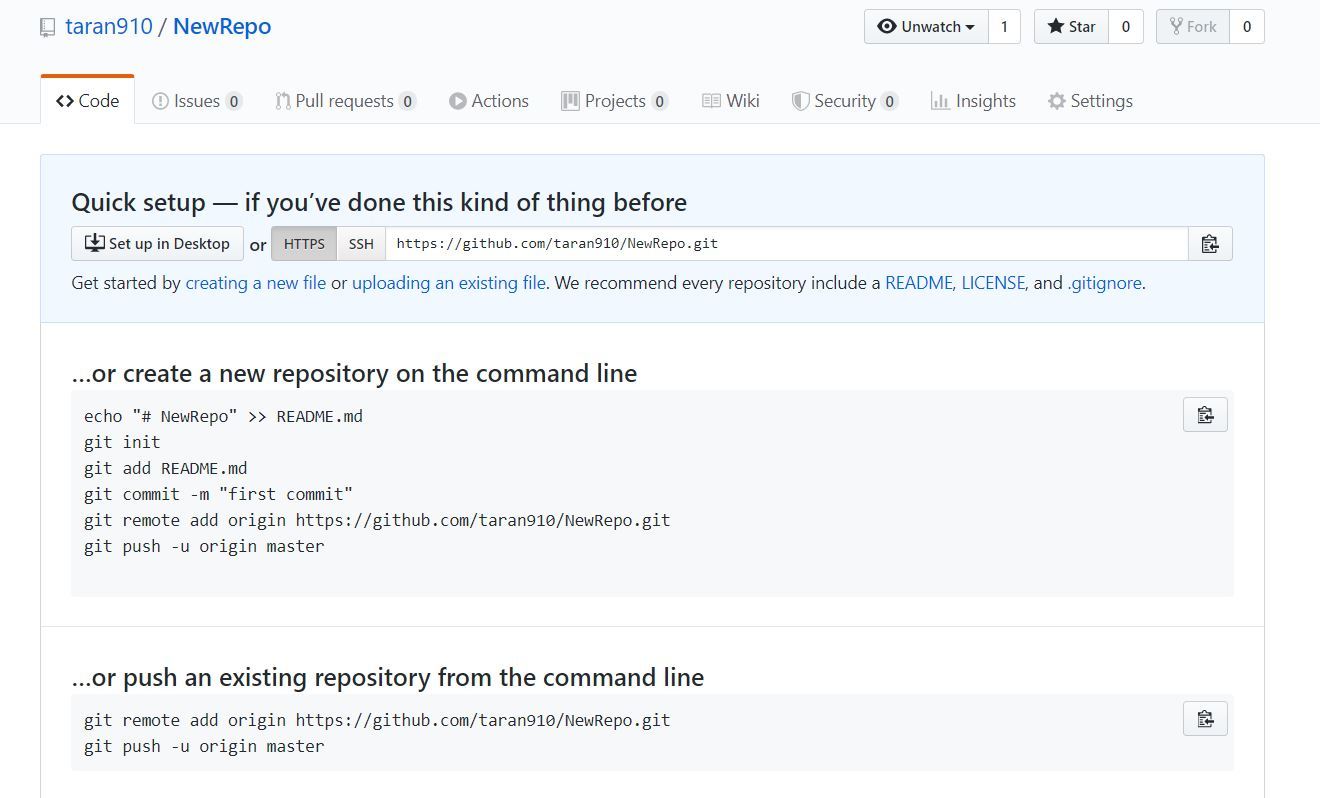
**Step 2:** Give a suitable name of your repository and create the repository



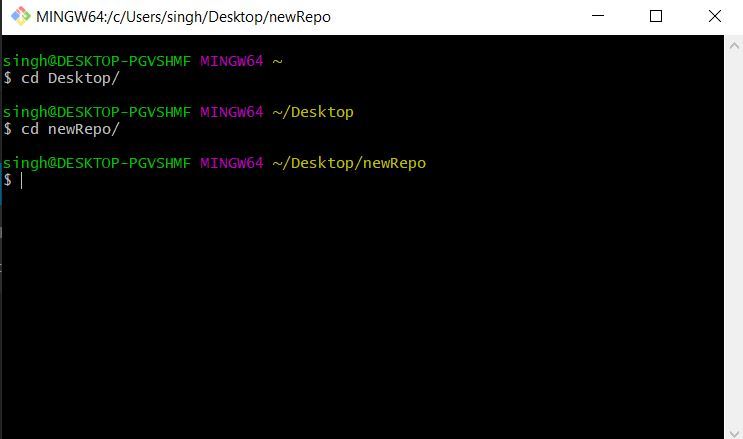
**Note:** You can choose to initialize your git repository with a README file, and further, you can mention your project details in it. It helps people know what this repository is about. However, it’s absolutely not necessary. But if you do initialize your repo with a README file using interface provided by GitHub, then your local repository won’t have this README file. So to avoid running into a snag while trying to push your files (as in step 3 of next section), after step 5 (where you initialize your local folder as your git repository), do following to pull that file to your local folder:

git pull

**Step 3:** The following will appear after creating the repository

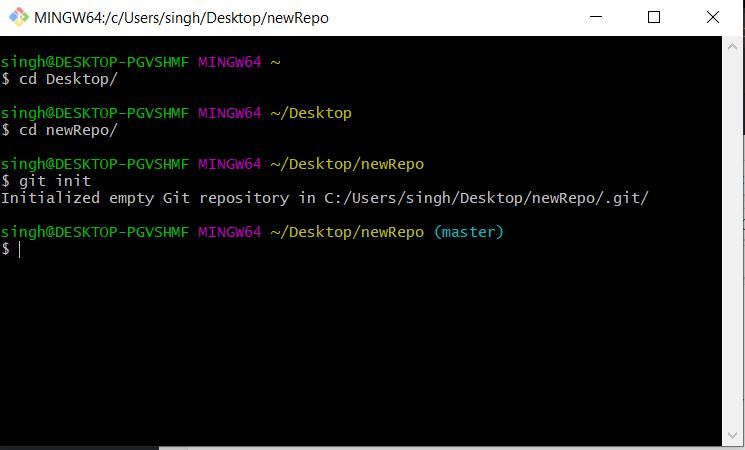


**Step 4:** Open Git Bash and change the current working directory to your local project by use of **cd** command.



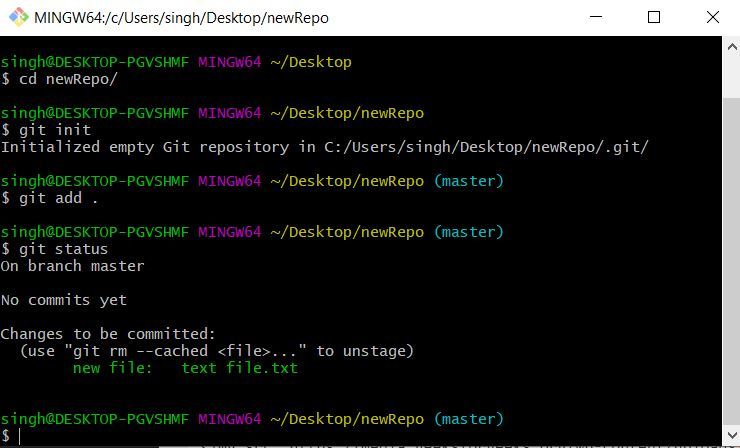
**Step 5:** Initialize the local directory as a Git repository.

git init



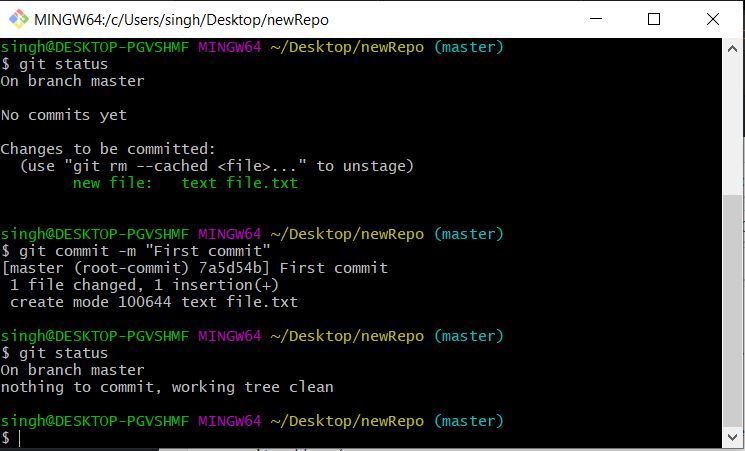
**Step 6:** Stage the files for the first commit by adding them to the local repository

git add .

**Step 7:** By **“git status”** you can see the staged files  


**Step 8:** Commit the files that you’ve staged in your local repository.

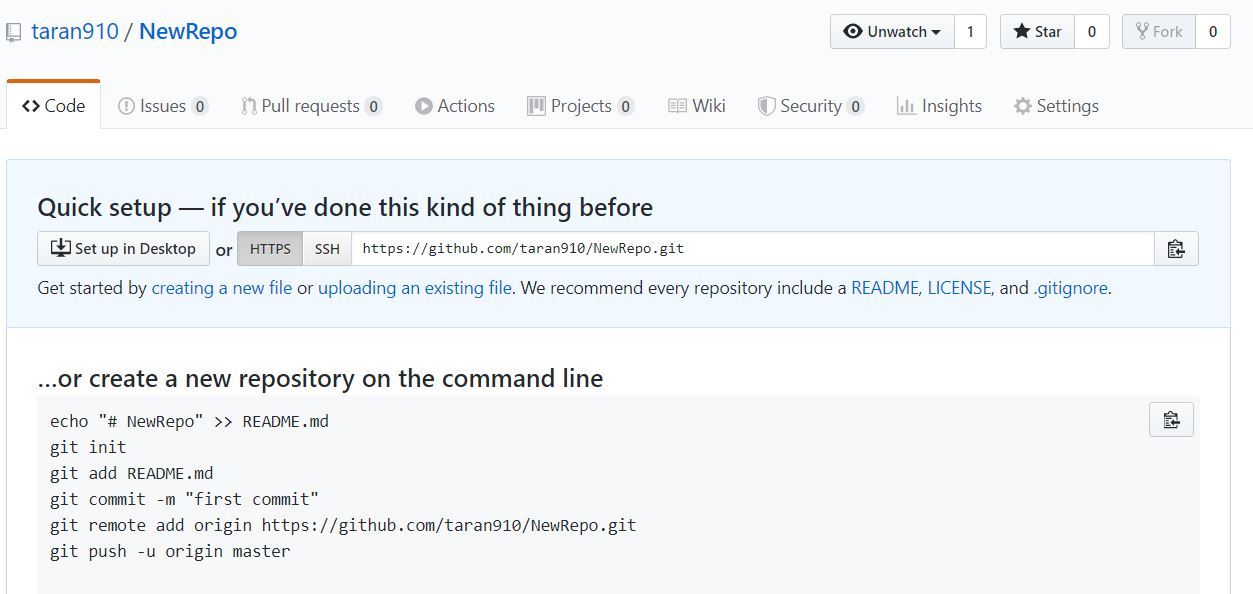
git commit -m "First commit"



Now After **“git status”** command it can be seen that nothing to commit is left, Hence all files have been committed.

**Push files to your Git repository**

**Step 1:** Go to Github repository and in **code** section **copy the URL**.

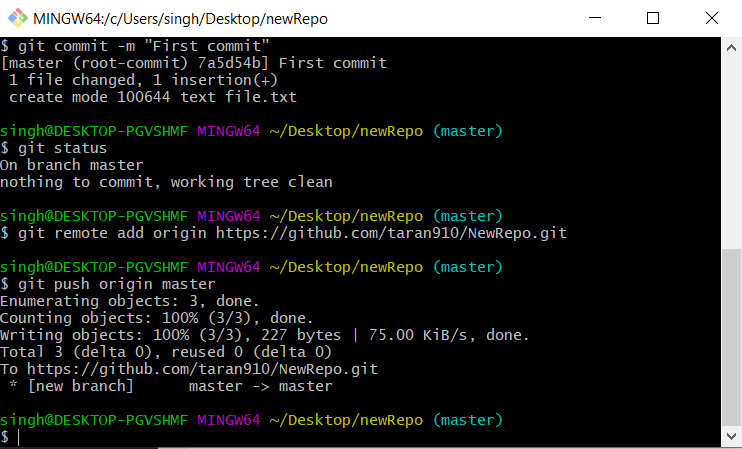


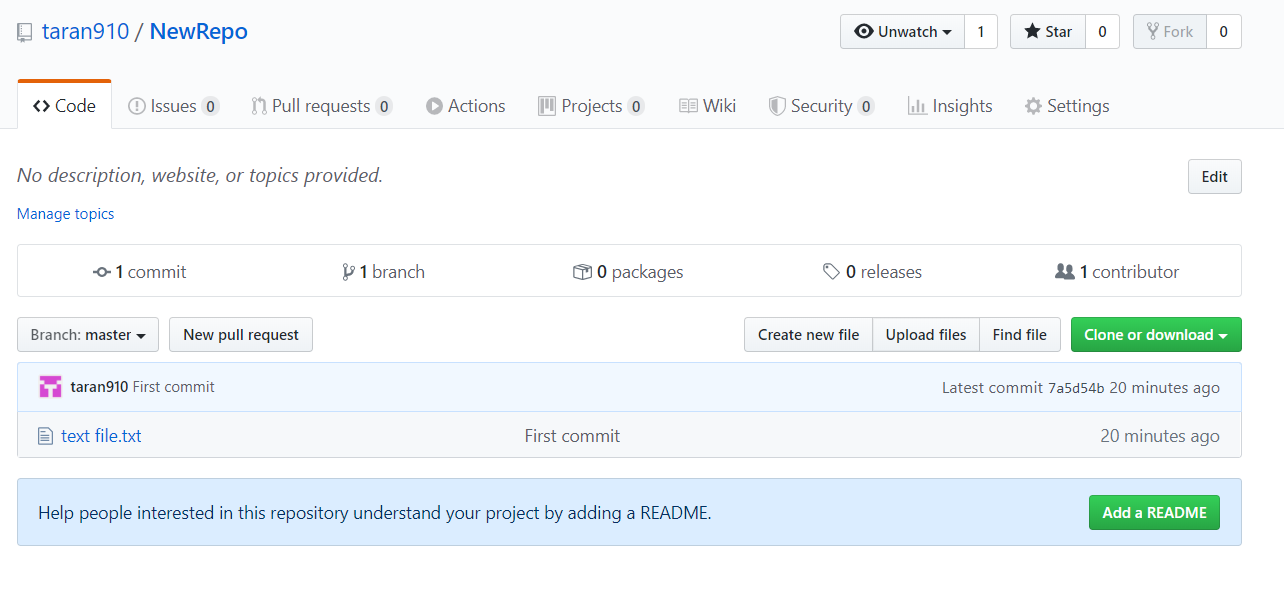
**Step 2:** In the Command prompt, add the URL for your repository where your local repository will be pushed.

git remote add origin repository\_URL

**Step 3:** Push the changes in your local repository to GitHub.

git push origin master

Here the files have been pushed to the master branch of your repository.  
  
Now in the GitHub repository, the pushed files can be seen.



**Saving changes to local repository**

Suppose the files are being changed and new files are added to local repository.  
To save the changes in the git repository:  
**Step 1:** Changes have to be staged for the commit.

git add .

or

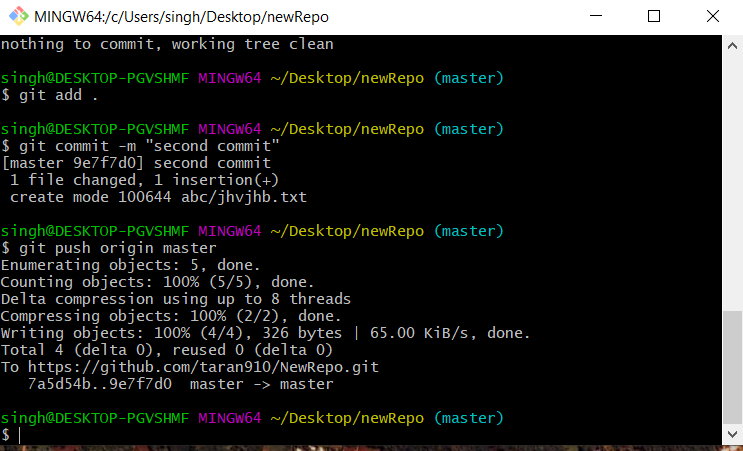
git add file\_name

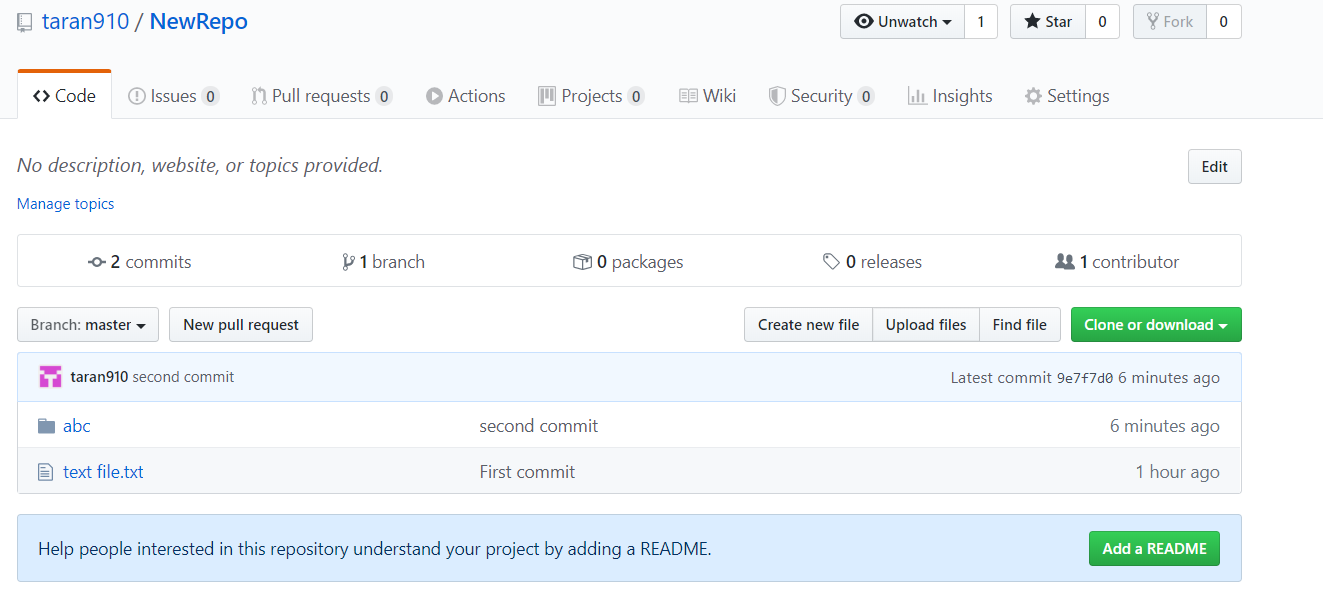
**Step 2:** Now commit the staged files.

git commit -m "commit\_name"

**Step 3:** Push the changes.

git push origin master



New changes can be seen  


**Branching through Git Bash**

**Branching in Github**

Suppose if a team is working on a project and a branch is created for every member working on the project.  
Hence every member will work on their branches hence every time the best branch is merged to the master branch of the project.  
The branches make it version controlling system and makes it very easy to maintain a project source code.

**Syntax:**

* List all of the branches in your repository.

git branch

* Create a new branch

git branch branch\_name

* Safe Delete the specified branch

git branch -d branch\_name

* Force delete the specified branch

git branch -D branch\_name

**Navigating between Branches**

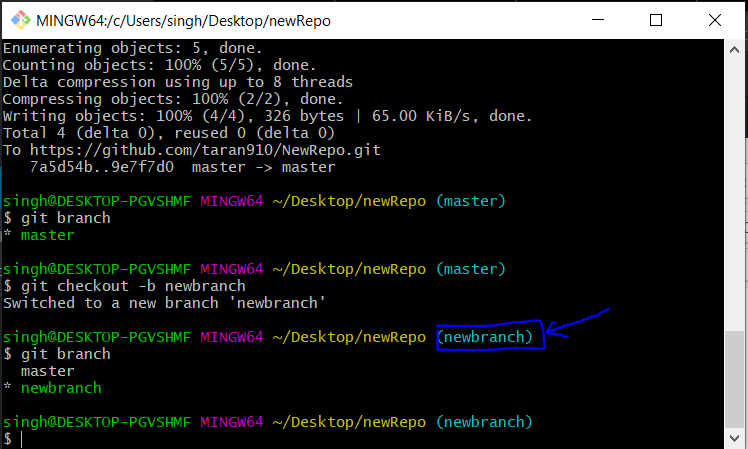
To navigate between the branches **git checkout** is used.

To create create a new branch and switch on it:

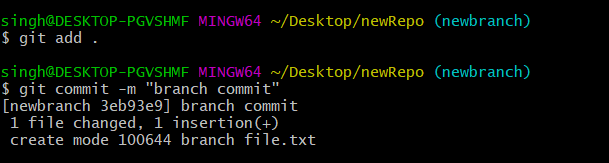
git checkout -b new\_branch\_name

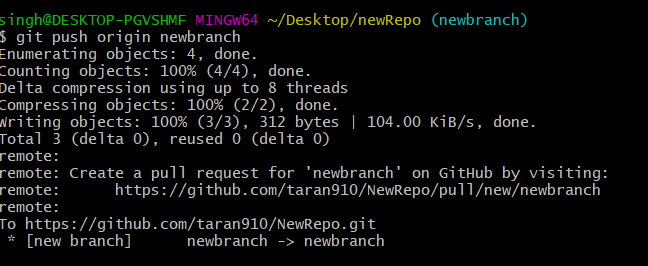
To simply switch to a branch

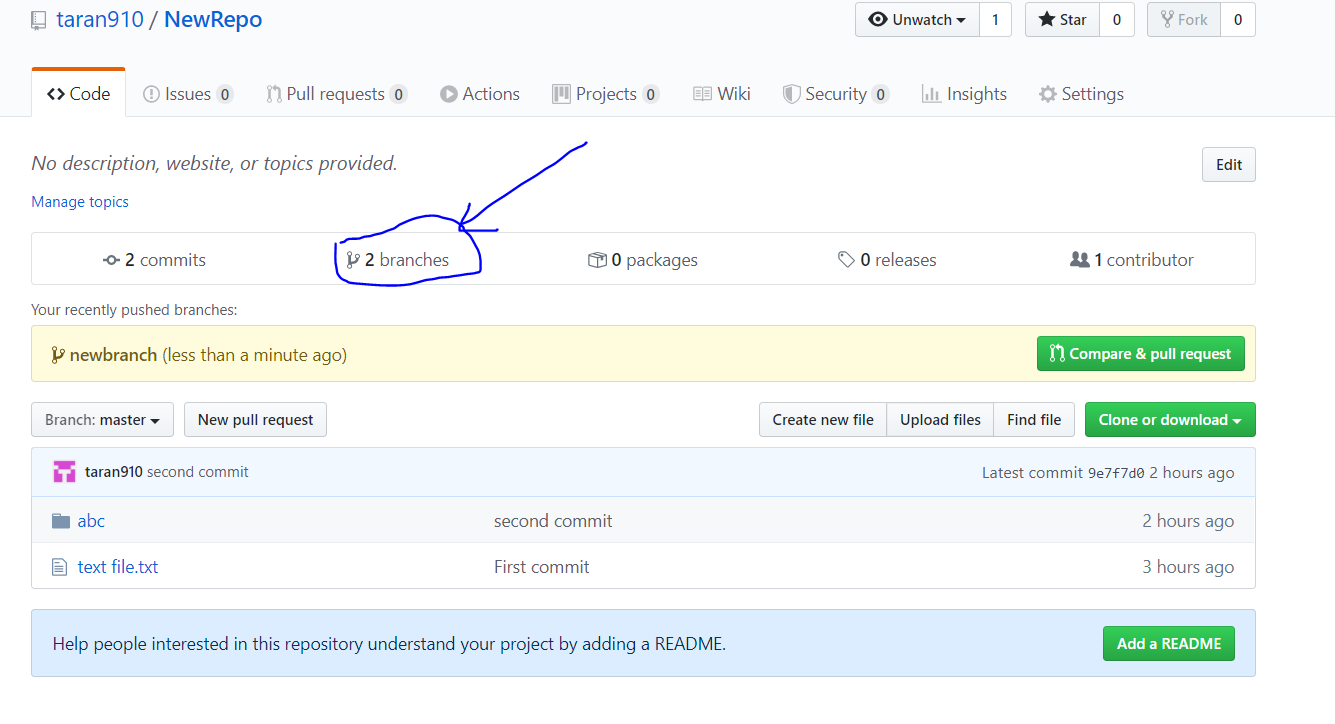
git checkout branch\_name

After checkout to branch you can see a \* on the current branch  


Now the same **commit add** and **commit actions** can be performed on this branch also.







**Merge any two branches**

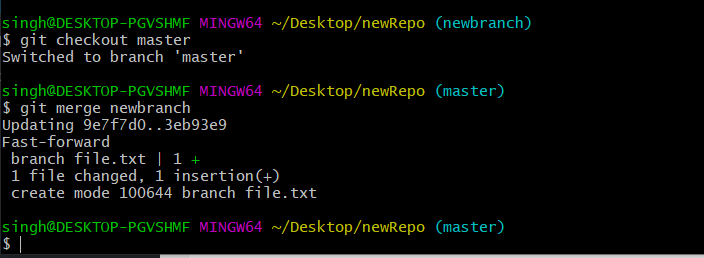
To merge a branch in any branch:

* First reach to the target branch

git checkout branch\_name

* Merge the branch to target branch

git merge new\_branch



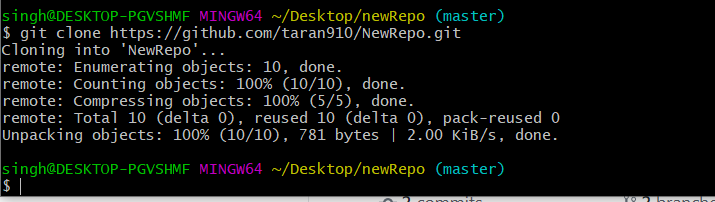
**Cloning Repository to system**

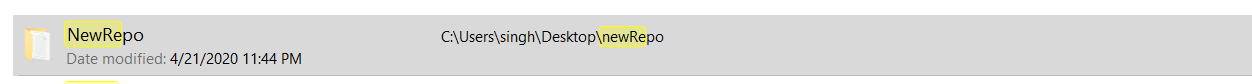
Cloning is used to get a copy of the existing git repository.  
When you run the **git clone** command it makes the zip folder saved in your default location

git clone url

This command saves the directory as the default directory name of the git repository  
To save directory name as your custom name an additional argument is to be passed for your custom name of directory

git clone url custom\_name





**Undoing commits**

When there is a situation when you forget to add some files to commit and want to undo any commit, it can be commit again using **--ammend**

**Syntax:**

git commit --amend

**Conclusion**

* To conclude it can be said that git bash is a command line platform which helps in enabling git and its elements in your system.
* There are a bunch of commands which are used in git bash.
* Git Bash is very easy to use and makes it easy to work on repositories and projects.