

Git

Git is a version control system for tracking changes in files and coordinating work among multiple users. It allows users to create and manage project history, work on different branches, and collaborate on code development. Its decentralized nature enables offline work and seamless synchronization.

GitHub

GitHub is a web-based platform that hosts Git repositories. It provides a collaborative environment for developers to store, manage, and share their code repositories. GitHub offers additional features like issue tracking, pull requests, and project management tools, making it a popular platform for open-source projects and team collaboration.

Git V/S GitHub

Git	GitHub
Distributed version control system.	Web-based hosting service for Git repositories.
Manages source code and version history locally on a developer's machine.	Hosts Git repositories in a centralized manner, providing collaboration and additional features.
Used by developers for local code management and version control.	Used by individuals, teams, and organizations for hosting and collaborating on Git repositories.

Git	GitHub
Branching, merging, committing, and version control.	Collaboration tools, issue tracking, pull requests, project management, and more.
Runs locally on a developer's machine.	Accessed via web browsers, offering a centralized and accessible repository hosting solution.
Command-line tools like Git Bash, Git CLI, and IDE integrations.	GitHub.com, GitHub Desktop, GitHub CLI, and other GitHub-related services.

Note :

Git is the underlying version control system, while GitHub is a platform built around Git, providing additional features and a web-based interface for hosting repositories.

Git Bash

Git Bash is a command-line interface (CLI) tool that provides a Unix-like shell environment on Windows systems. It combines the power of Git, a distributed version control system, with the functionality of a Unix shell. Git Bash allows you to interact with Git repositories and perform various version control operations through the command line.

Here are a few key points about Git Bash:

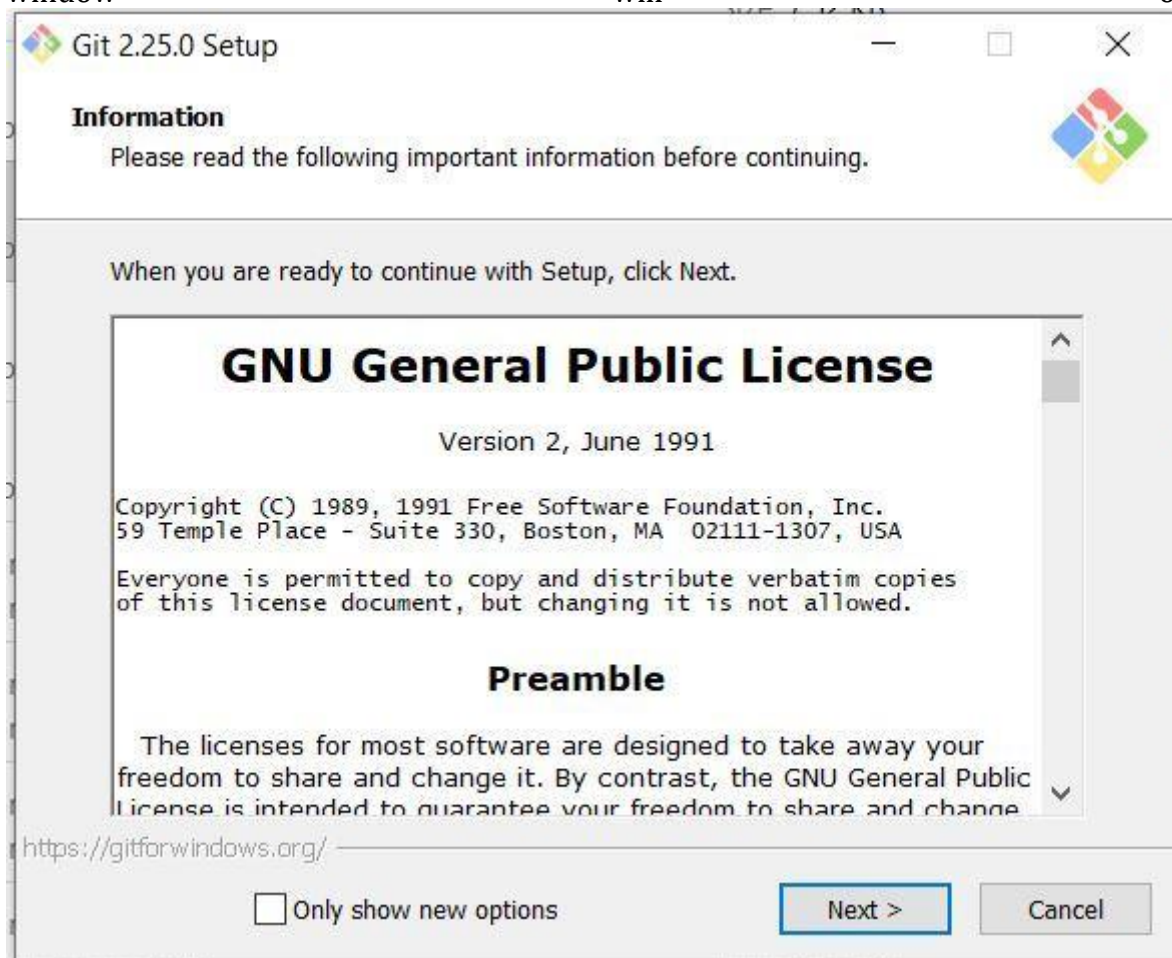
- *Terminal Emulator:* Git Bash provides a terminal emulator for Windows, which allows you to execute commands and interact with the underlying operating system.
- *Unix-like Shell:* Git Bash uses the Bash shell, which is a popular shell environment commonly found in Unix-like systems. It provides a set of commands and features that enable you to navigate directories, manipulate files, and run scripts.
- *Git Integration:* Git Bash comes bundled with Git, so you have access to all the Git commands and features. You can initialize Git repositories, clone remote repositories,

create branches, stage and commit changes, merge branches, and perform various other Git operations.

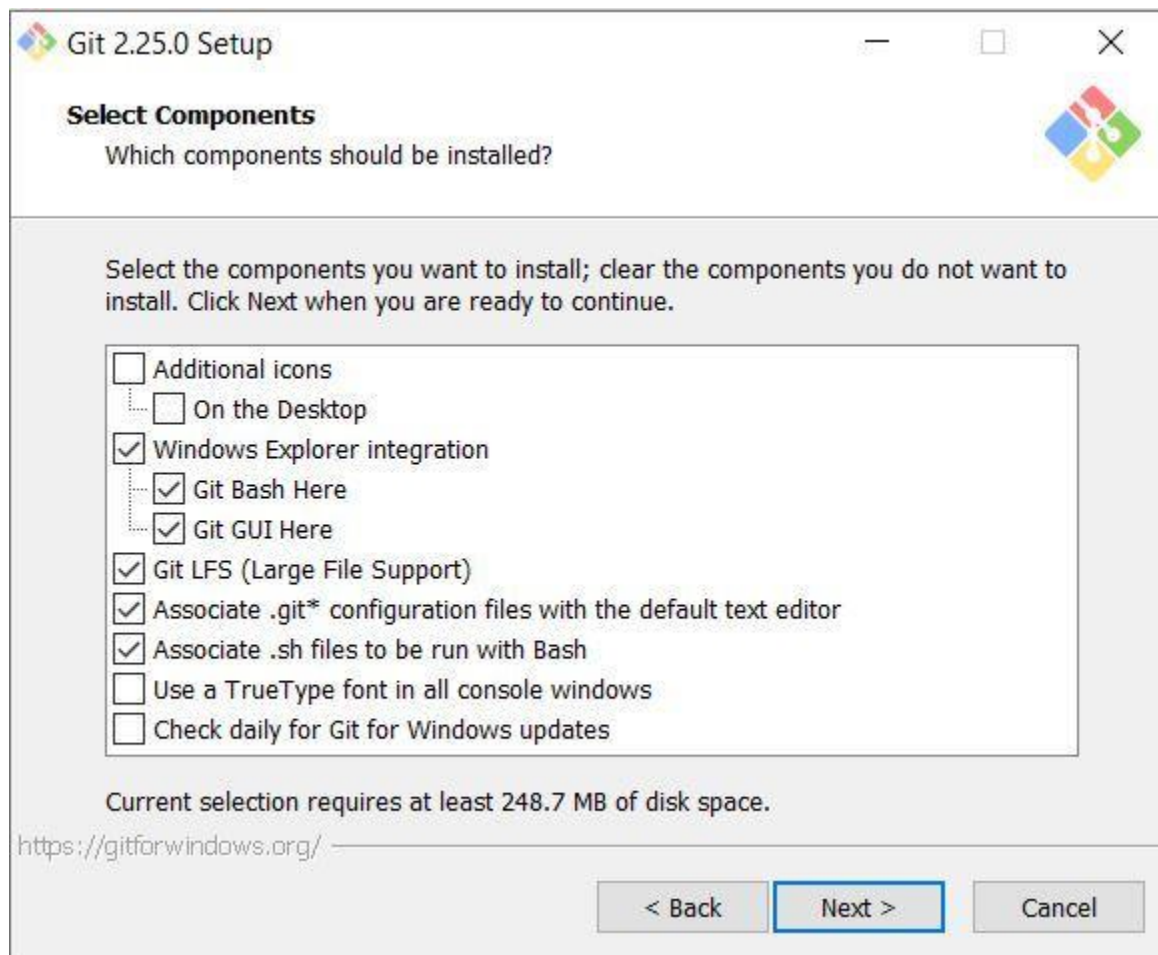
- *Compatibility:* Git Bash is designed to work seamlessly on Windows systems, providing a consistent experience for Git users familiar with Unix-like environments. It supports Git commands and workflows, allowing you to work with Git repositories on Windows just as you would on Linux or macOS.
- *Additional Tools:* Git Bash also includes additional Unix utilities and tools commonly used in shell environments, such as grep, awk, sed, and more. These tools enhance your productivity and enable you to perform complex operations and scripting tasks.

Installation Of Git Bash

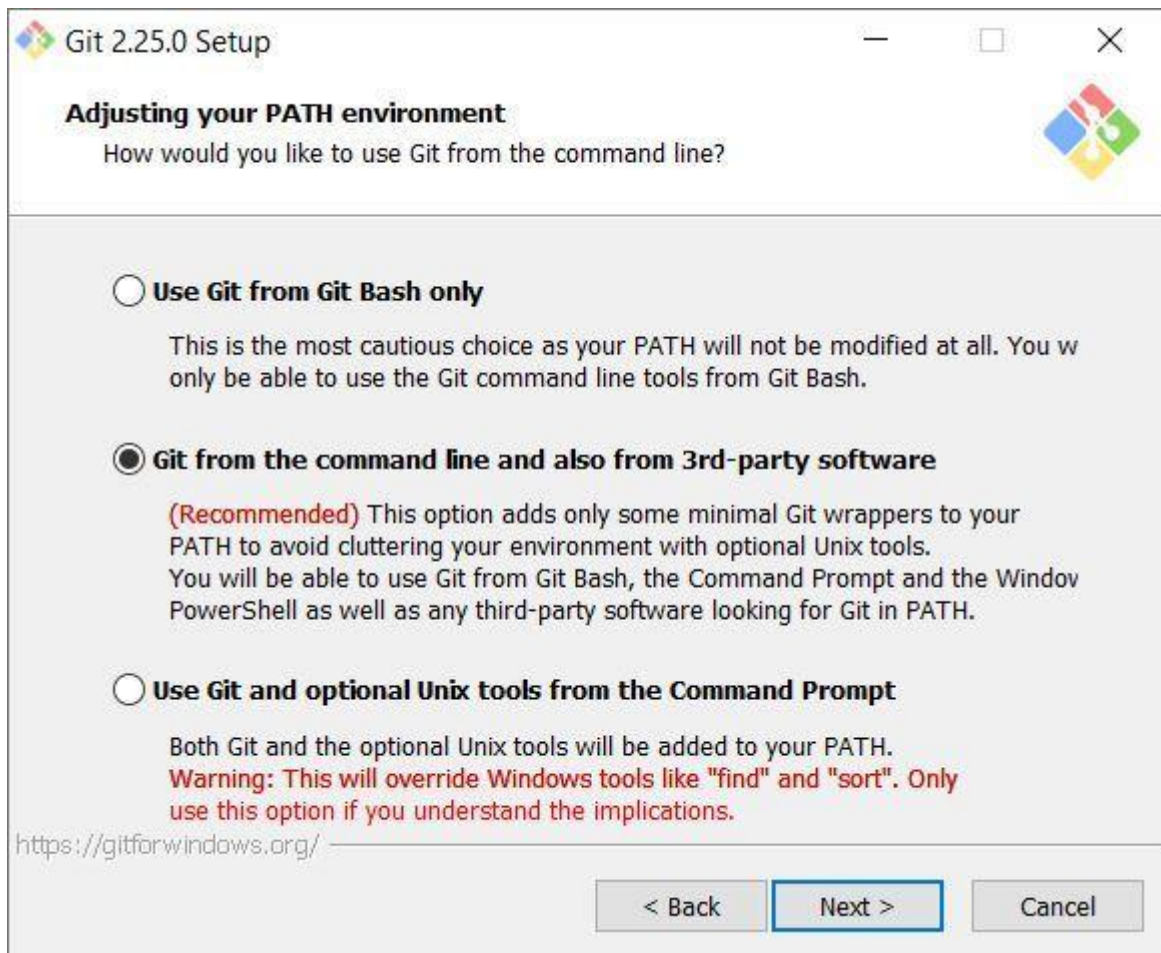
Step 1: The .exe file installer for Git Bash can be downloaded from <https://gitforwindows.org/> Once downloaded execute that installer, following window will occur:-



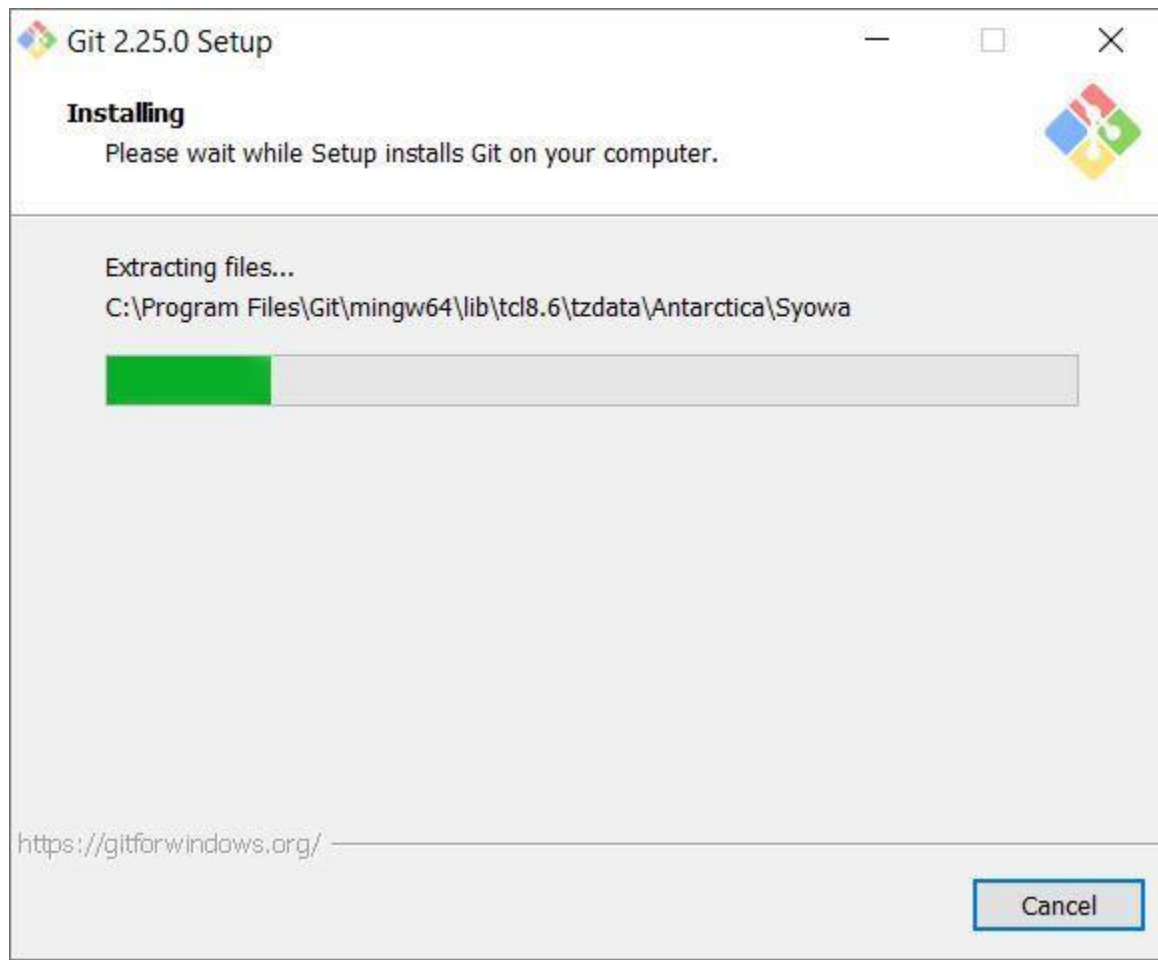
Step 2: Select the components that you need to install and click on the Next button.



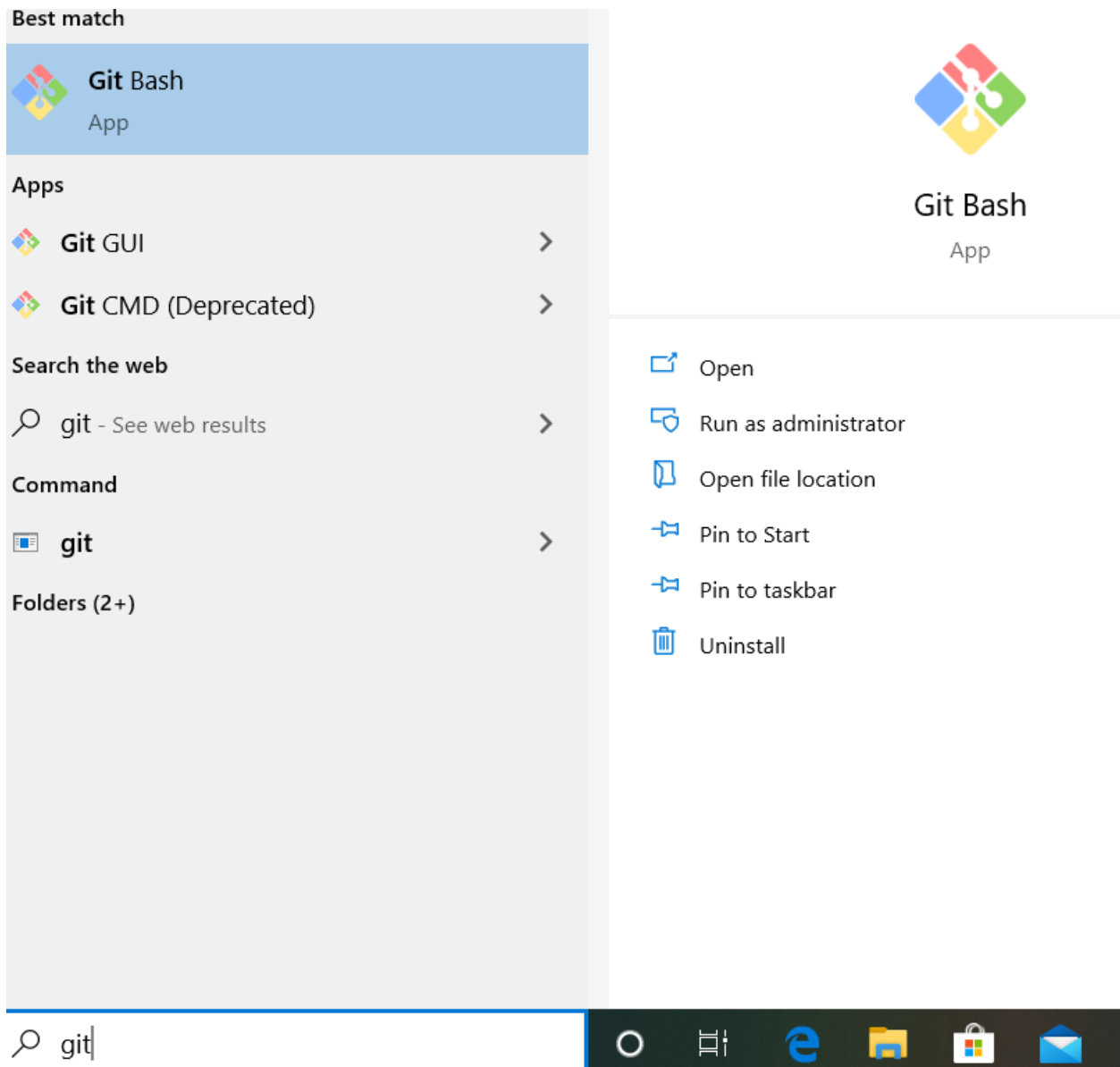
Step 3: Select how to use the Git from command-line and click on Next to begin the installation process.



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



Step 5 : 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"
```

SSH Keys

- SSH keys are a pair of cryptographic keys used for secure communication between a client and a server.
- They consist of a public key and a private key.
- The public key is stored on the server, while the private key is kept by the client.
- SSH keys provide a more secure and convenient alternative to password-based authentication.
- They are used for secure remote access to servers, secure file transfers, and other SSH-based services.

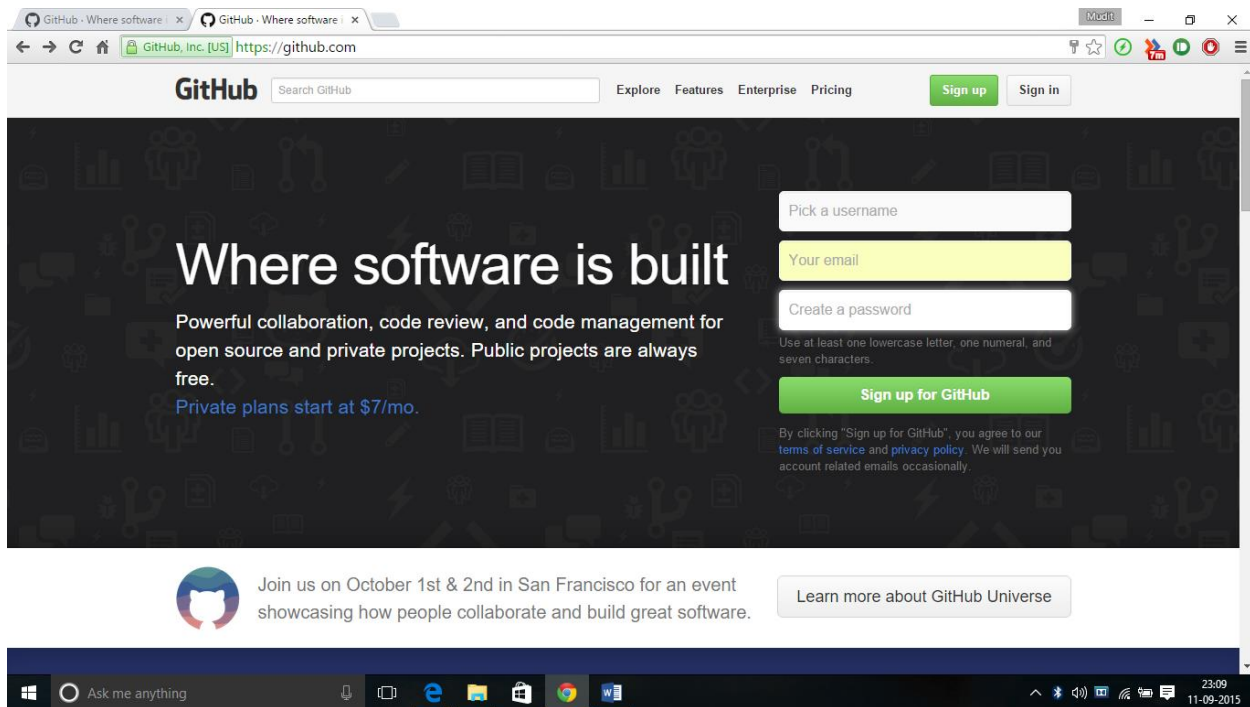
Generating New SSH Keys :

1. Open a terminal or command prompt.
2. Run the SSH key generation command: **ssh-keygen -t rsa -b 2048 -C "your_email@example.com"**
3. Optionally, you can provide a custom file path and name for the keys.
4. You'll be prompted to enter a passphrase for the key. It's recommended to use a strong passphrase for added security.
5. The SSH key pair (public and private key) will be generated and saved to the specified file path or the default location.
6. Retrieve the public key using **cat ~/.ssh/id_rsa.pub** or by opening the public key file.
7. Copy the public key and add it to your SSH key settings on the relevant platform (e.g., GitHub, GitLab) or provide it to the server administrator.

Setting Up GitHub

1. Creating a GitHub Account :

Go to github.com and enter the required user credentials asked on the site and then click on the SignUp for GitHub button.



2. Verify email:

After signing up, you'll receive an email from GitHub to verify your email address. Click on the verification link provided in the email to complete the process.

3. Configure Git:

Open Git Bash or your preferred terminal and configure your Git username and email using the following commands:

```
git config --global user.name "Your Name"
git config --global user.email "youremail@example.com"
```

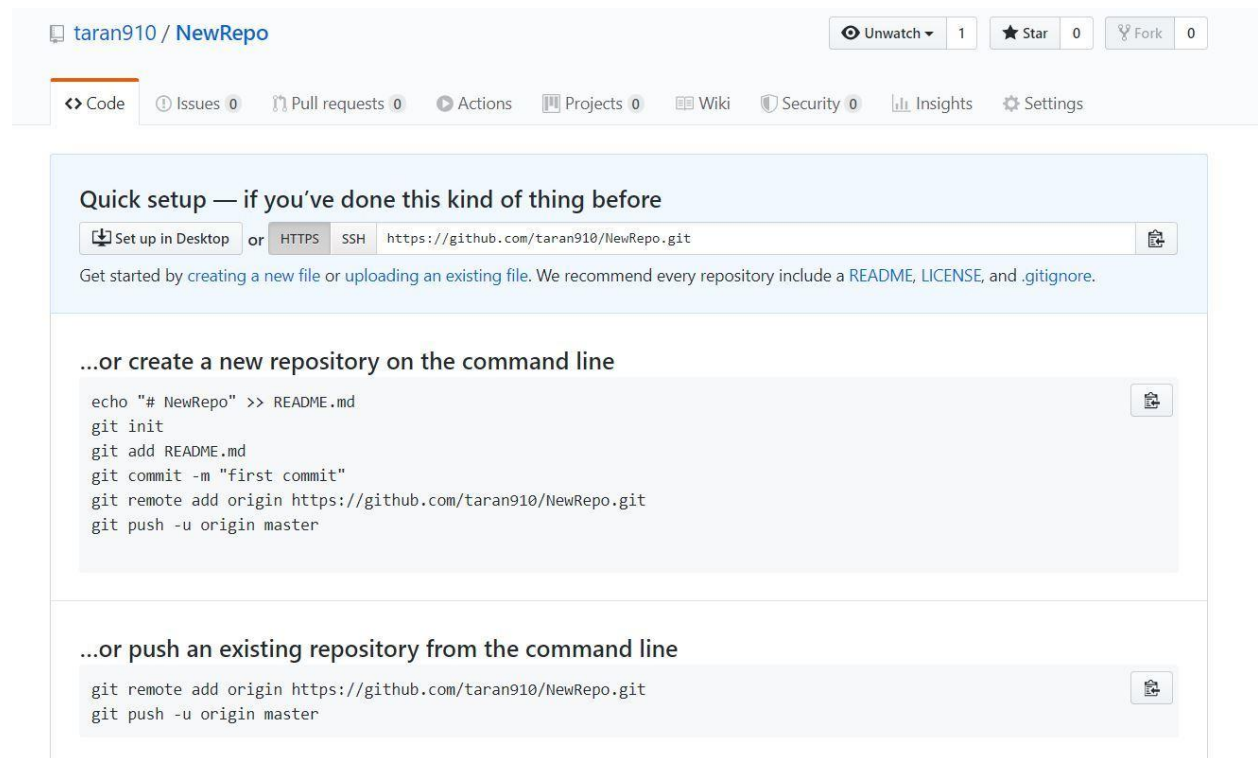
4. Generate SSH keys:

Generate SSH keys on your local machine. Make sure to add the public key to your GitHub account.

5. Create a repository:

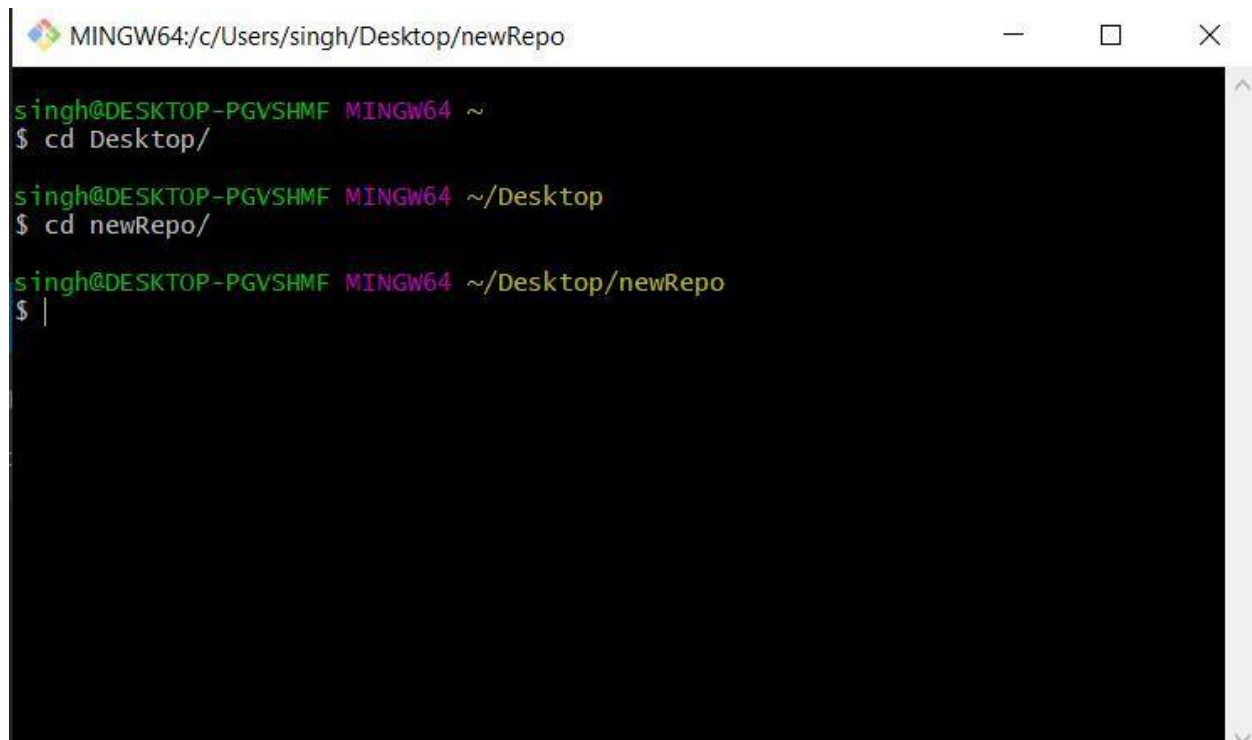
Once you're signed in to your GitHub account, you can create a new repository by clicking on the "New" button on the main page or navigating to the repositories section and selecting

"New repository". Provide a name for your repository and choose the visibility (public or private).



6. Opening Git Bash

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

A screenshot of a Windows command prompt window. The title bar at the top reads "MINGW64:/c/Users/singh/Desktop/newRepo" and includes standard window controls (minimize, maximize, close). The command prompt shows a user named "singh" at a machine named "DESKTOP-PGVSHMF" in a "MINGW64" environment. The user has navigated from their home directory (~) to the Desktop directory using "cd Desktop/", and then to a subdirectory named "newRepo" using "cd newRepo/". The prompt is currently at the "newRepo" directory, indicated by the path "~/Desktop/newRepo" and a vertical cursor after the "\$" prompt.

```
MINGW64:/c/Users/singh/Desktop/newRepo

singh@DESKTOP-PGVSHMF MINGW64 ~
$ cd Desktop/

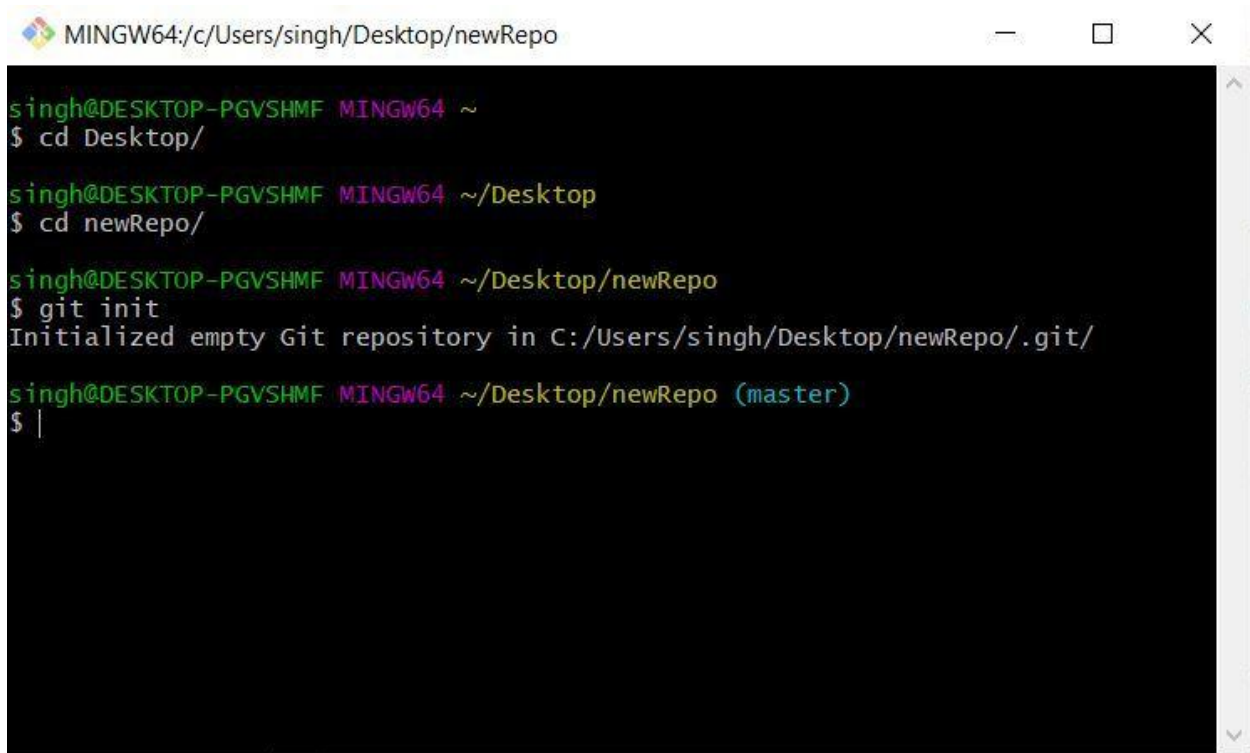
singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop
$ cd newRepo/

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo
$ |
```

7. Set up a local Git repository:

Initialize the local directory as a Git repository.

```
git init
git remote add origin git@github.com:your-username/your-repo.git
```

A screenshot of a Windows command prompt window titled "MINGW64:/c/Users/singh/Desktop/newRepo". The window has standard Windows window controls (minimize, maximize, close) in the top right corner. The command prompt shows the following sequence of commands and output:

```
singh@DESKTOP-PGVSHMF MINGW64 ~  
$ cd Desktop/  
  
singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop  
$ cd newRepo/  
  
singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo  
$ git init  
Initialized empty Git repository in C:/Users/singh/Desktop/newRepo/.git/  
  
singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)  
$ |
```

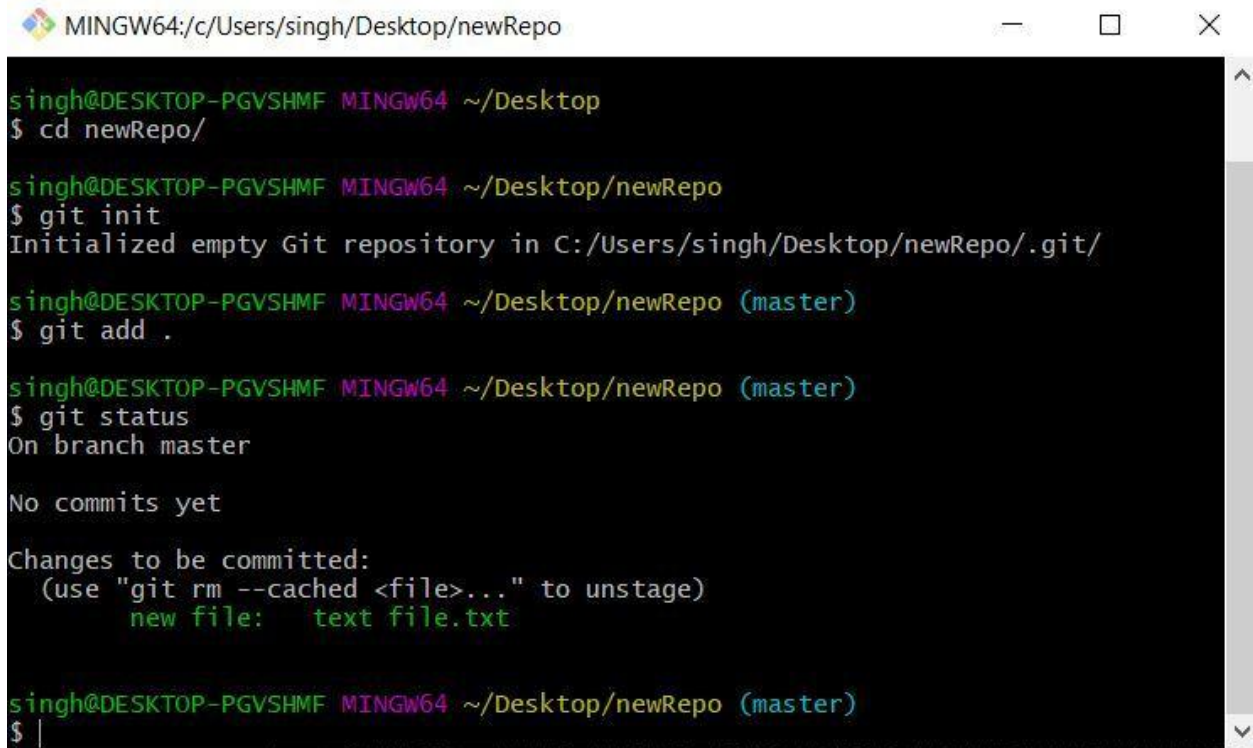
8. Create and add files:

Add files to your local repository by creating them or copying existing files into the repository directory. Use the following command to add files to the staging area:

```
git add
```

9. Check Status:

Using **"git status"** we can see the staged files



```
MINGW64:/c/Users/singh/Desktop/newRepo
singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop
$ cd newRepo/

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo
$ git init
Initialized empty Git repository in C:/Users/singh/Desktop/newRepo/.git/

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git add .

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git status
On branch master

No commits yet

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

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ |
```

10. Commit changes:

Commit the changes to your local repository with a descriptive message using the following command:

```
git commit -m "Your commit message"
```

```
MINGW64:/c/Users/singh/Desktop/newRepo
singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git status
On branch master

No commits yet

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

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git commit -m "First commit"
[master (root-commit) 7a5d54b] First commit
 1 file changed, 1 insertion(+)
 create mode 100644 text file.txt

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$ git status
On branch master
nothing to commit, working tree clean

singh@DESKTOP-PGVSHMF MINGW64 ~/Desktop/newRepo (master)
$
```

11. Push to GitHub:

Push your local repository to GitHub using the following command:

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

Note:

Testing SSH Connection

1. Ensure that you have the SSH client installed on your system (such as OpenSSH).
2. Open a terminal or command prompt.
3. Run the SSH connection command: **ssh -T git@github.com**
 - o Replace **git@github.com** with the appropriate SSH URL for the server you want to connect to.
4. If this is your first time connecting to the server, you may see a message asking you to confirm the server's authenticity. Type "yes" to proceed.
5. If you have set up SSH keys correctly and they are properly configured on the server, you will see a success message or a welcome message from the server.
6. If the connection is not successful, ensure that your SSH keys are set up correctly and that you have provided the correct SSH URL for the server.

Marked as Read
Report An Issue