

**Source Code Management Lab**

**Name: Aroor Akshay Rao**

**Branch: B.Tech**

**Section: CSE 4th Sem**

**Course: Source Code Management**

**SEN: A86605223028**

**Date of Submission: 6th June 2025**



Amity School of   
Engineering & Technology

**GIT INSTALLING**

**Step 1: Downloading Git**

1. Open your web browser and navigate to the official Git website: <https://git-scm.com>.
2. On the homepage, you will see a "**Download**" button that automatically detects your OS. Click on the "Download" button to download the appropriate installer for your operating system (Windows, macOS, or Linux).
3. Alternatively, you can manually select your OS from the website to download a specific version.

A screenshot of a website

AI-generated content may be incorrect.

**Step 2: Running the Git Installer**

Locate the downloaded Git.exe file and double-click to run it.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 3: License (Terms and Conditions)**

Read the **GNU** General Public License’s terms and conditions and click on **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 4: Choose Installation Location**

Choose the installation location (default is **C:\Program Files\Git**) and click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 5: Select the Components**

Select the components you want (default options are fine) and click **Next**.

A screenshot of a computer program

AI-generated content may be incorrect.

**Step 6: Select Start Menu Folder**

Choose the Start Menu folder where Git shortcuts will be placed. By default, the folder is named **"Git"**. Keep the default name and click **Next** to Proceed.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 7: Choose the Text Editor**

Choose a default text editor (select **Vim**) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 8: Adjusting Initial Branch Name**

Choose the default name for the first branch when initializing a new Git repository. Go with **‘Let Git Decide’** option setting the branch as **Master** branch and proceed with **Next**.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Step 9: Adjusting PATH Environment**

Select **Git from the command line and also from third-party software** (recommended). Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 10: Choosing the SSH Executable**

Select "**Use bundled OpenSSH**" for better compatibility and Click on **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 11: Choosing the HTTP Transport Background**

Choose Use the **OpenSSL library** (default) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 12: Configuring Line Ending Configs**

Select **Checkout Windows-style, commit Unix-style line endings** (recommended) and Click **Next**.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Step 13: Configuring the Terminal Emulator**

Select **Use MinTTY** **(default terminal for MSYS2**) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 14: Choosing the Default Behaviour**

Select **Fast-forward or Merge** (recommended) option and click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 15: Choosing a Credential Helper**

Select **Git Credential Manager** (recommended) and Click **Next**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 16: Configuring Extra Options**

Select **Enable file system caching** (recommended) and Click on **Install**.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 17: Installation Overview**

A progress bar (**green bar**) will appear, indicating that Git is being installed. Wait for the installation to complete. This may take a few minutes.

A screenshot of a computer

AI-generated content may be incorrect.

**Step 18: Completing the Git Set - Up Wizard**

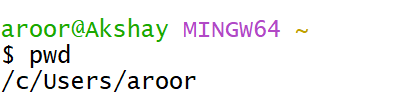
Once the installation is complete, **"Completing the Git Setup Wizard"** screen appears. Check the ‘**Launch Git bash’** option and Click on **Finish**.

A screenshot of a computer

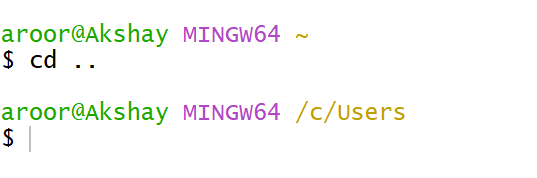
AI-generated content may be incorrect.

**COMMANDS IN GIT**

1. **pwd**
   * *Purpose*: Displays the Present Working Directory.
   * *Usage*: Use this to find the directory you're currently working in.



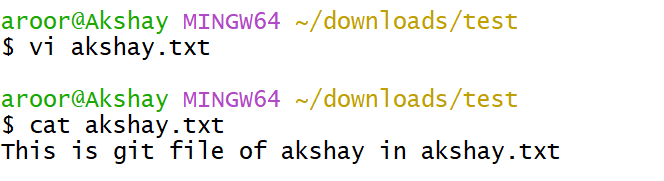
1. **cd**
   * *Purpose*: Change directory.
   * *Usage*: Navigate to a different directory



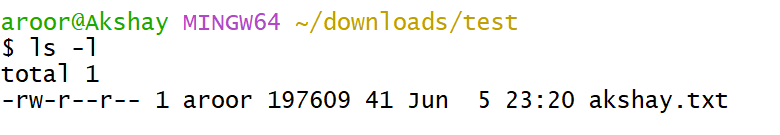
1. **ls**
   * *Purpose*: Lists all folders in the current directory.



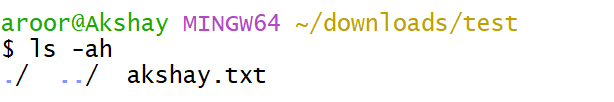
1. **vi**
   * *Purpose*: Open the Vim/Vi editor to create or edit files.
   * *Usage*: vi filename. Press I to insert, esc to exit insert mode, and :wq to save and quit.
2. **cat**
   * *Purpose*: View file contents.
   * *Usage*: cat filename.



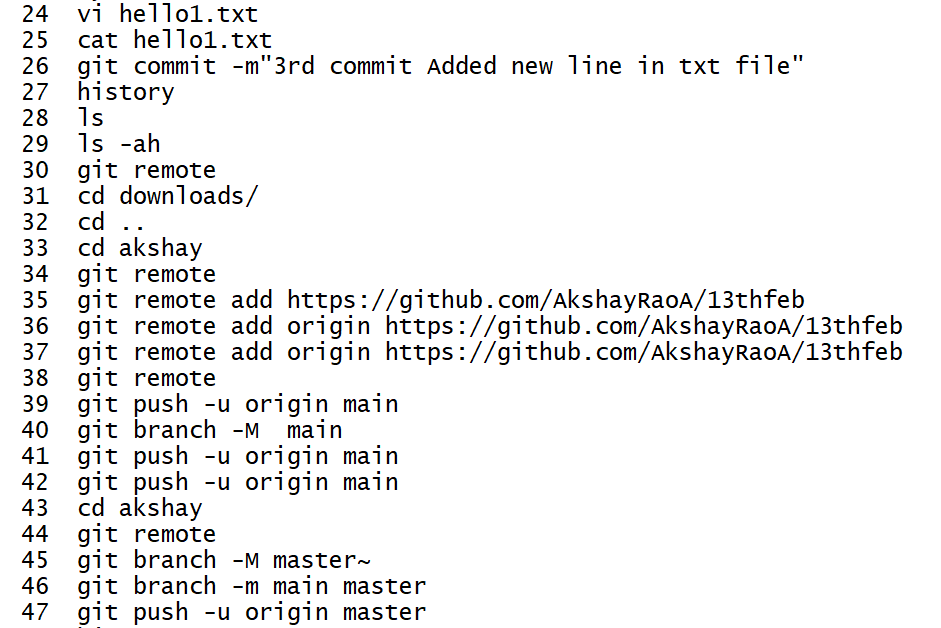
1. **ls -l**
   * *Purpose*: Lists files in long format with additional details (permissions, owners, size).



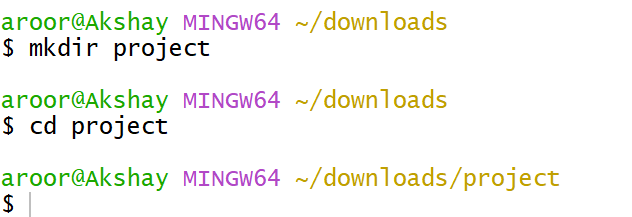
1. **ls -ah**
   * *Purpose*: Lists all files, including hidden ones.



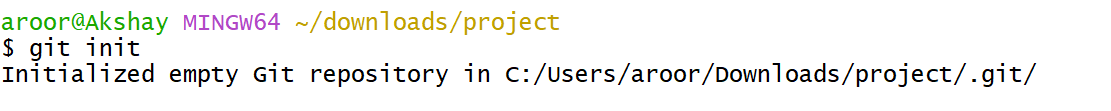
1. **history**
   * *Purpose*: Displays the history of previously run commands.



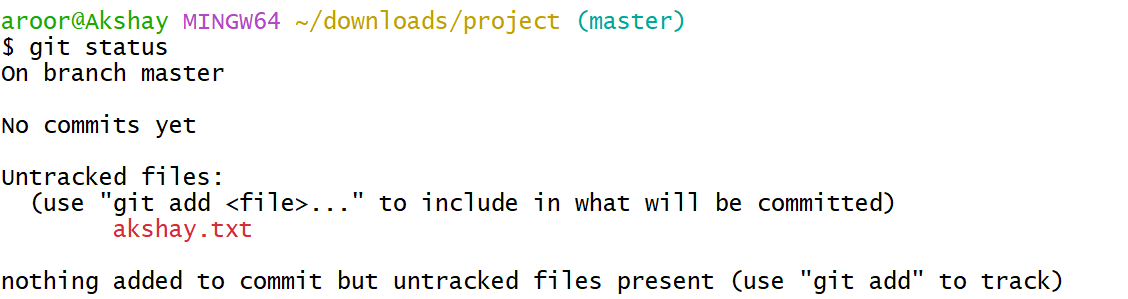
1. **mkdir**
   * *Purpose*: Create a new folder.
   * *Usage*: mkdir foldername.



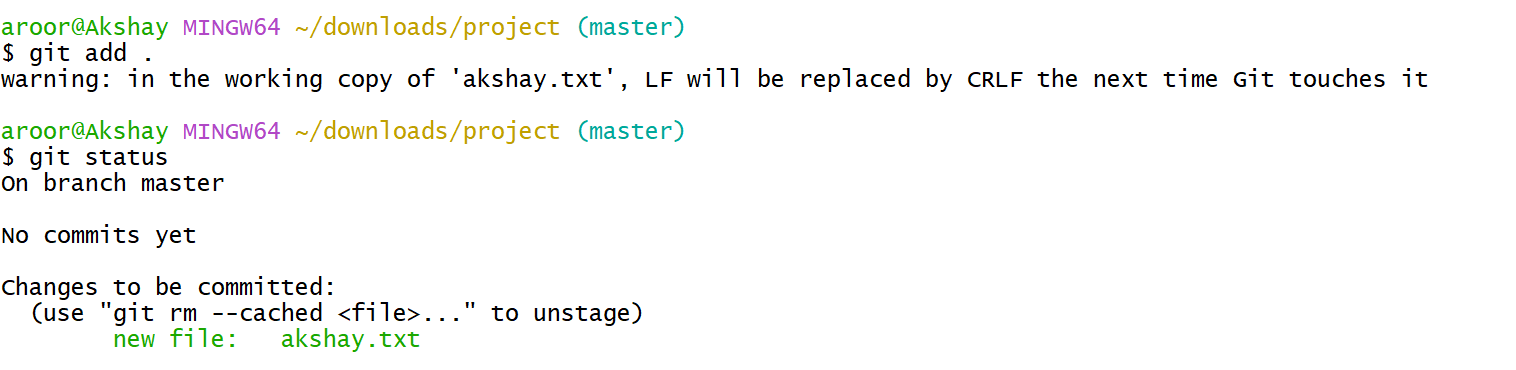
1. **git init**
   * *Purpose*: Initializes an empty Git repository in the current directory.



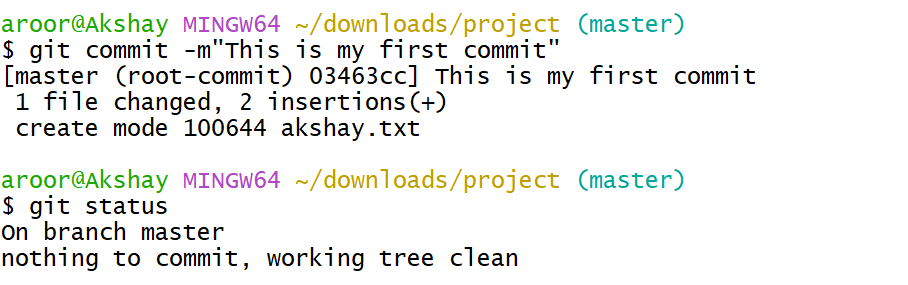
1. **git status**
   * *Purpose*: Displays the current status of the working directory and staging area, showing:
   * Modified files not yet staged.
   * Staged files ready for commit.
   * Untracked files.
   * Current branch and whether it's up-to-date with the remote branch.



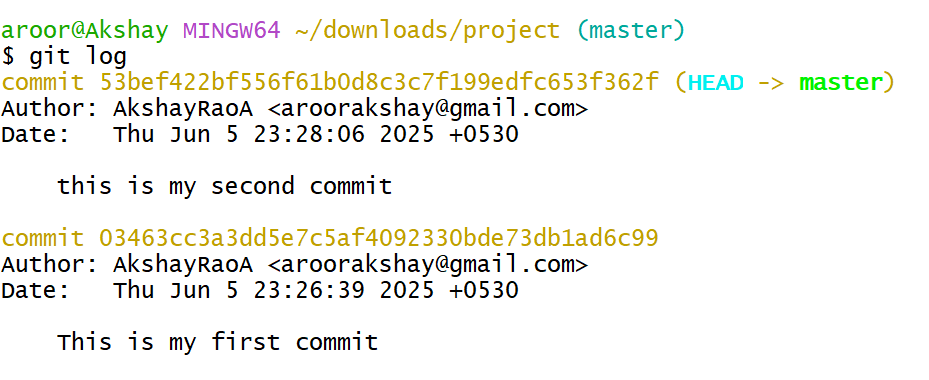
1. **git add**
   * *Purpose*: Stage a file to prepare it for commit.
   * *Usage*: git add filename.



1. **git commit**
   * *Purpose*: Commit changes to the repository with a message.
   * *Usage*: git commit -m "Message".

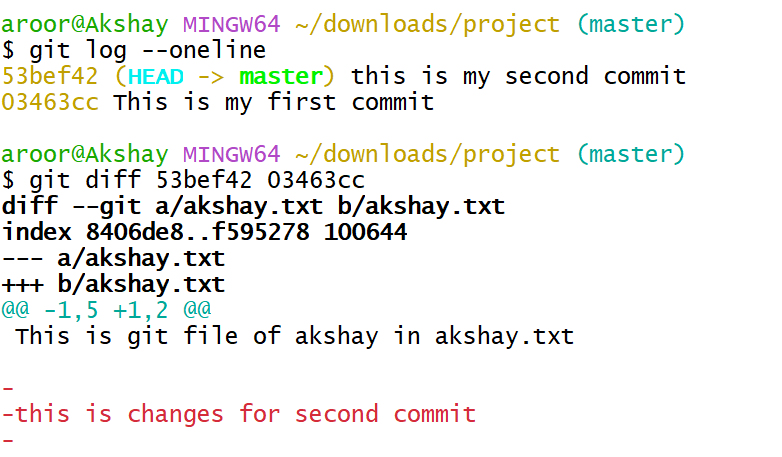


1. **git config**
   * *Purpose*: Configure Git with username and email.
   * *Usage*:
     + git config --global user.mail "email@example.com"
     + git config --global user.name "Your Namve"
2. **git log**
   * *Purpose*: View commit history.



1. **git diff**
   * *Purpose*: shows the changes made in the working directory since the last commit.
2. **git –oneline**

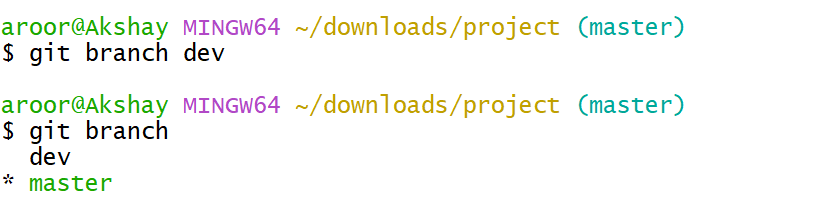
*Purpose:* View a concise commit history



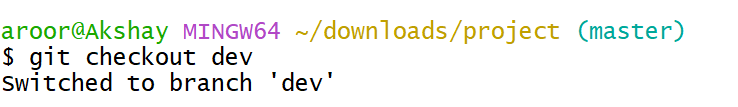
1. **git branch <branch name>**
   * *Purpose*: Creates a new branch.

**git branch**

* + *Purpose*: Lists all branches in the repository.



1. **git checkout <branch name>**
   * *Purpose*: Switch to the master branch.
   * *Usage*:



1. **git remote add <url>**

* *Purpose:* Connects your local repository to a remote repository.

1. **git remote**

* *Purpose:* Lists the short names of all configured remote repositories.

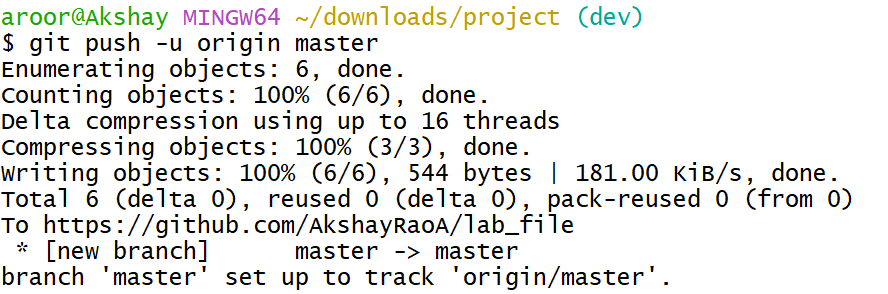
1. **git remote -v**

* Purpose: Displays the URLs of all remote repositories linked to your local repository, showing both **fetch** and **push** directions.

**

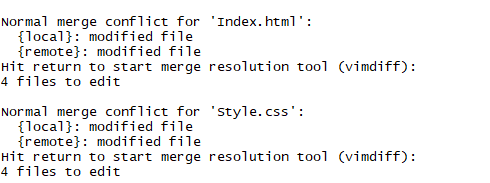
1. **git push**

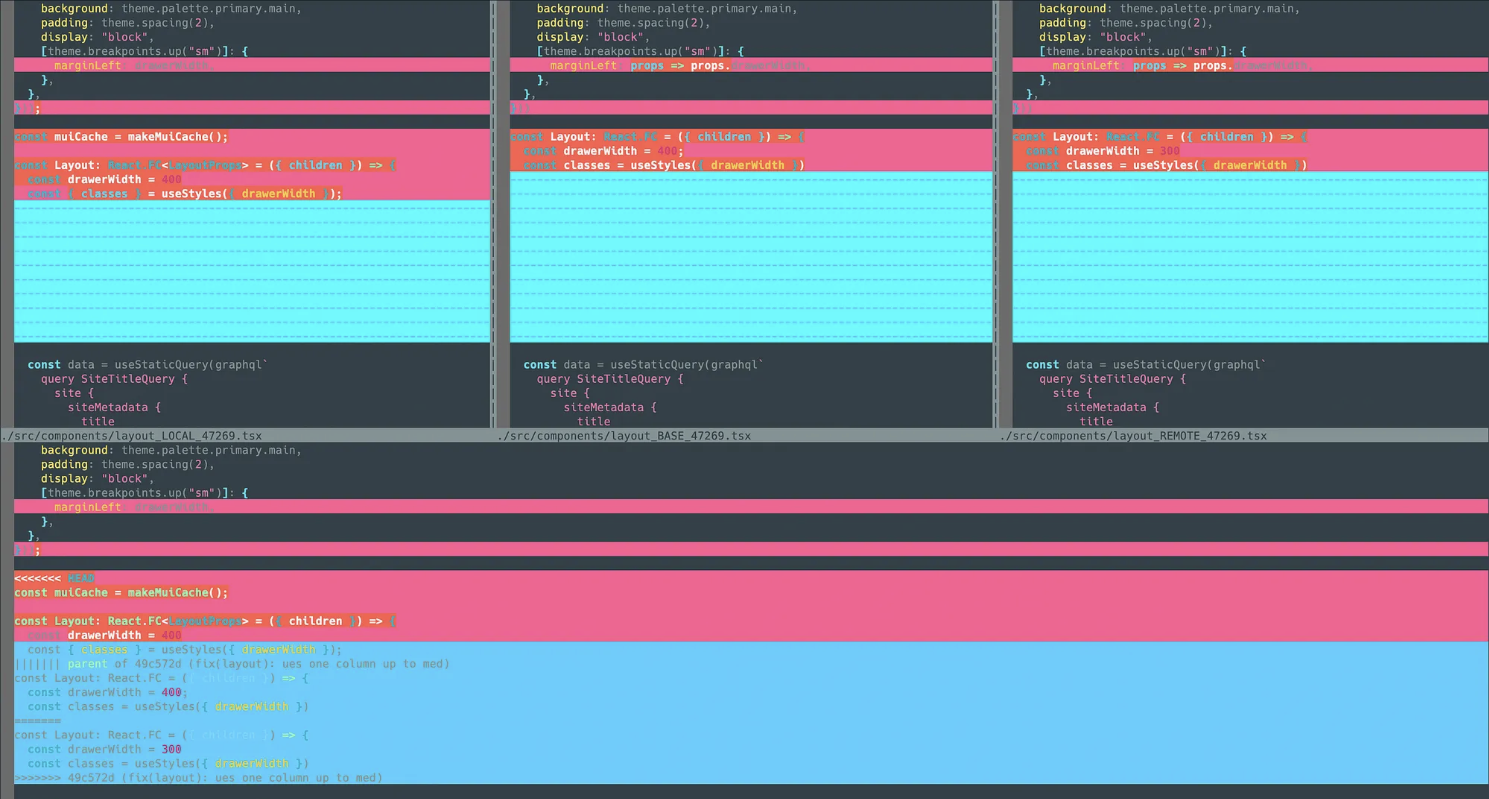
* Purpose: Pushes the local main branch to the remote repository named origin



**GIT MERGE**

* **Production Code** is the finalized, deployable version of software used in a live environment.
* **Branches** in Git allow multiple versions of code to exist simultaneously, starting from the project's inception.
* **Merging** happens when different branches need to be combined. Git does not automatically decide what to keep—it requires manual resolution.
* **Conflict Resolution**: If conflicts arise, Git will halt the merge. A merge tool helps resolve differences.

**Handle a simple merge conflict**

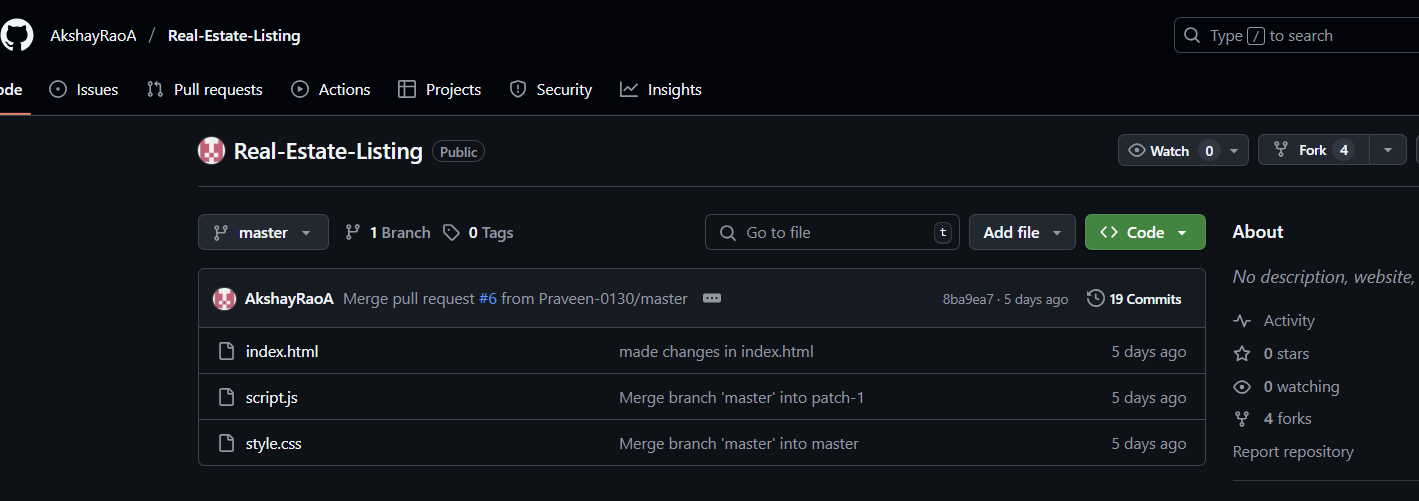


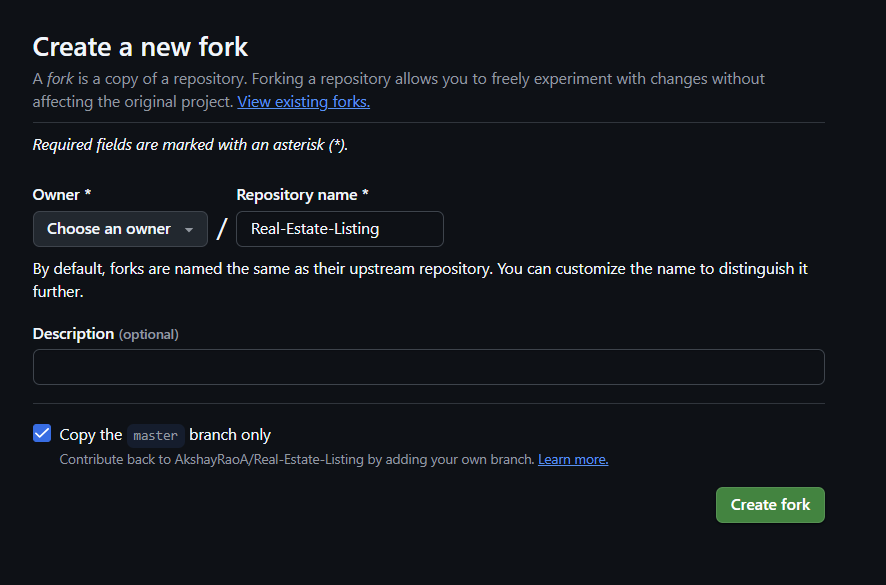
**GIT IGNORE**



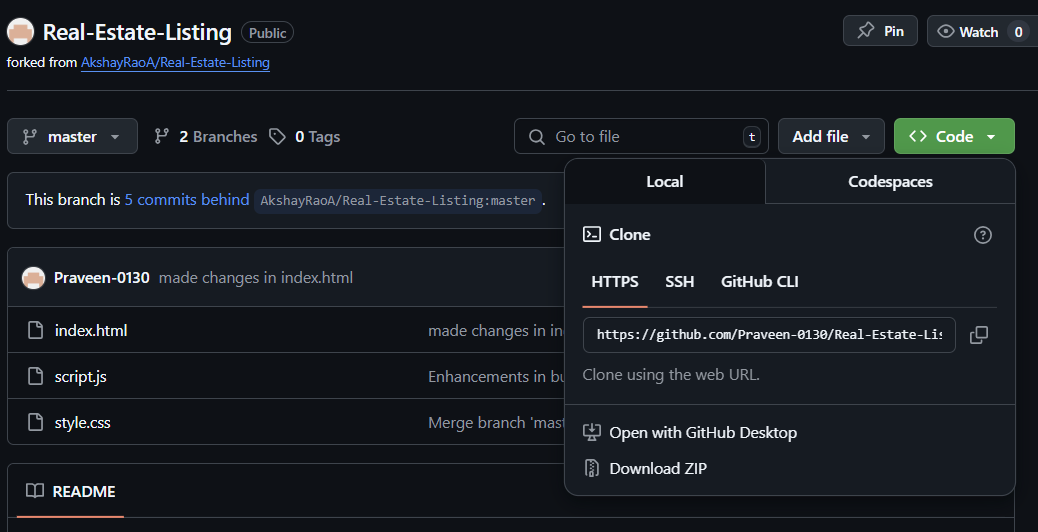
**GIT CLONE**

Navigate a project that you are interested in and fork it

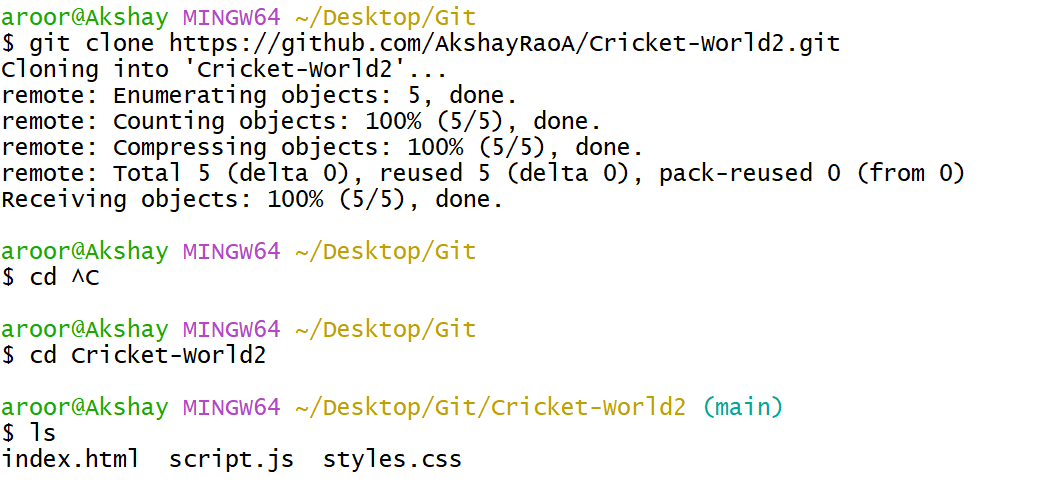




Copy The URL

****

Go to git bash



Now the file can be accessed