

Subject Name: Source Code Management

Subject Code: CS181

Cluster: Beta

Department: DCSE



Department of Computer Science & Engineering

Chitkara University Institute of Engineering and Technology, Punjab

Jan- June
(2021-22)



Institute/School Name	Chitkara University Institute of Engineering and Technology		
Department Name	Department of Computer Science & Engineering		
Programme Name	Bachelor of Engineering (B.E.), Computer Science & Engineering		
Course Name	Source Code Management	Session	2021-22
Course Code	CS181	Semester/Batch	2nd/2021
Vertical Name	Beta	Group No	G01
Course Coordinator	Dr. Neeraj Singla		
Faculty Name	Dr. Monit Kapoor		

Name: Abhinav

Roll No:2110990045

Date: April 8, 2022

Table of Content

S. No.	Title	Page No.
1	Version Control with Git	I
2	Objective	II
3	Installation Of Git	1 - 5
4	Setting Up GitHub Account	6
5	Configuration of GIT	7-8
6	Program to Generate Logs	9 – 12
7	Create and Visualize Branches	13-16
8	GIT Lifecycle Description	17 - 18
9	Uploading Data on GitHub	19-22
10	Final project	23



1. Version Control with GIT

Version control systems are software tools that help software teams manage changes to source code overtime. As development environments have accelerated, version control systems help software teams work faster and smarter.

A version Control system records all the changes made to a file or set of files, so a specific version may be called later if needed.

How version control helps high performing development and DevOps teams prosper?

1. Version Control Systems (VCS) have seen great improvements over the past few decades and some are better than others.
2. VCS are sometimes known as SCM (Source Code Management) tools or RCS (Revision Control System).
3. One of the most popular VCS tools in use today is called Git. Git is a Distributed VCS, a category known as DVCS, more on that later. Like many of the most popular VCS systems available today, Git is free and open source.



2. Objective of GIT

Git is a version control system used for tracking changes in computer files. It is generally used for source code management in software development.

- Git is used to tracking changes in the source code
- The distributed version control tool is used for source code management
- It allows multiple developers to work together
- It supports non-linear development through its thousands of parallel branches

Features of Git =>

- Tracks history
- Free and open source
- Supports non-linear development
- Creates backups
- Scalable
- Supports collaboration
- Branching is easier
- Distributed development

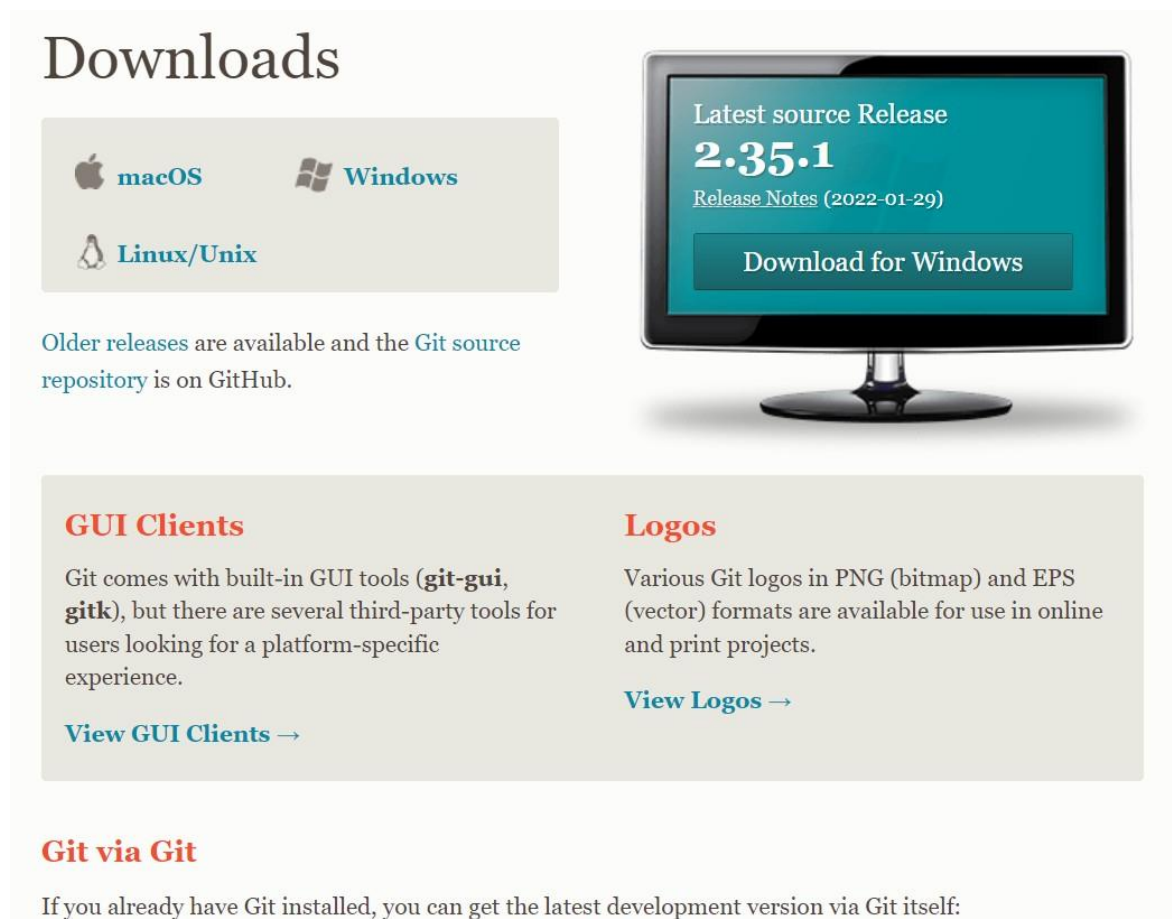
3. Installation of GIT

Step 1 =>

To download the Git installer, visit the Git official site and go to the download page.

The link for the download page is <https://git-scm.com/downloads>

The page looks like as: -

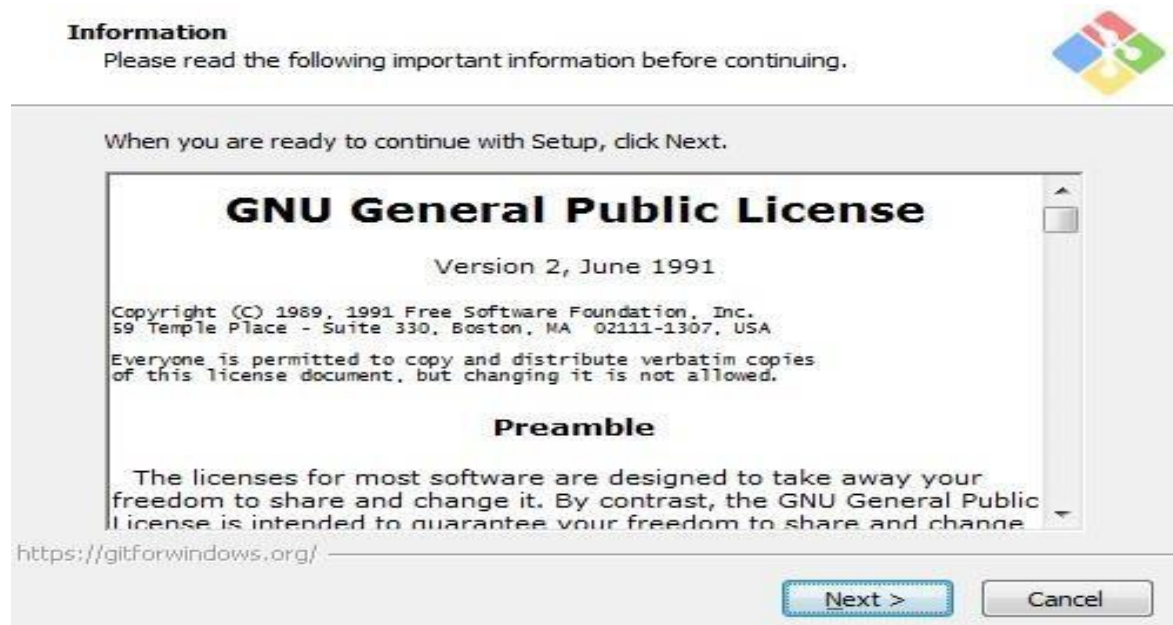


Click on the package given on the page as **download 2.23.0 for windows**. The download will start after selecting the package.

Now, the Git installer package has been downloaded.

Step 2 =>

Click on the download installer file and then click on next.
The page looks like as: -



Step 3 =>

Simply click on the next button as it automatically selects the required file.
The page looks like as: -

Select Components
Which components should be installed?

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

☐ Additional icons

- ☐ On the Desktop

☒ Windows Explorer integration

- ☒ Git Bash Here
- ☒ Git GUI Here

☒ Git LFS (Large File Support)

☒ Associate .git* configuration files with the default text editor

☒ Associate .sh files to be run with Bash

☐ Use a TrueType font in all console windows

☐ Check daily for Git for Windows updates

Current selection requires at least 253.0 MB of disk space.

<https://gitforwindows.org/>

< Back Next > Cancel

Step 4 =>

You can choose your preferred choice. Click next to continue.
The page looks like as: -

Adjusting your PATH environment
How would you like to use Git from the command line?

☐ **Use Git from Git Bash only**
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.

☒ **Git from the command line and also from 3rd-party software**
(Recommended) This option adds only some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from Git Bash, the Command Prompt and the Windows PowerShell as well as any third-party software looking for Git in PATH.

☐ **Use Git and optional Unix tools from the Command Prompt**
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only use this option if you understand the implications.

<https://gitforwindows.org/>

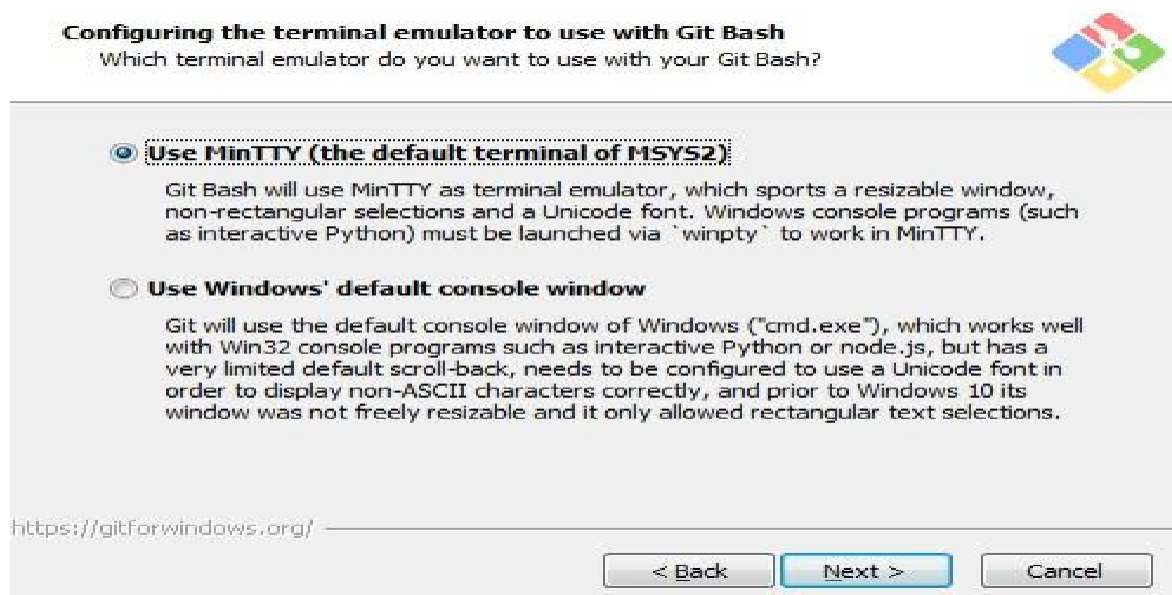
< Back Next > Cancel

Step 5 =>

Note: - Just simply click on next as it automatically selects the

required file.

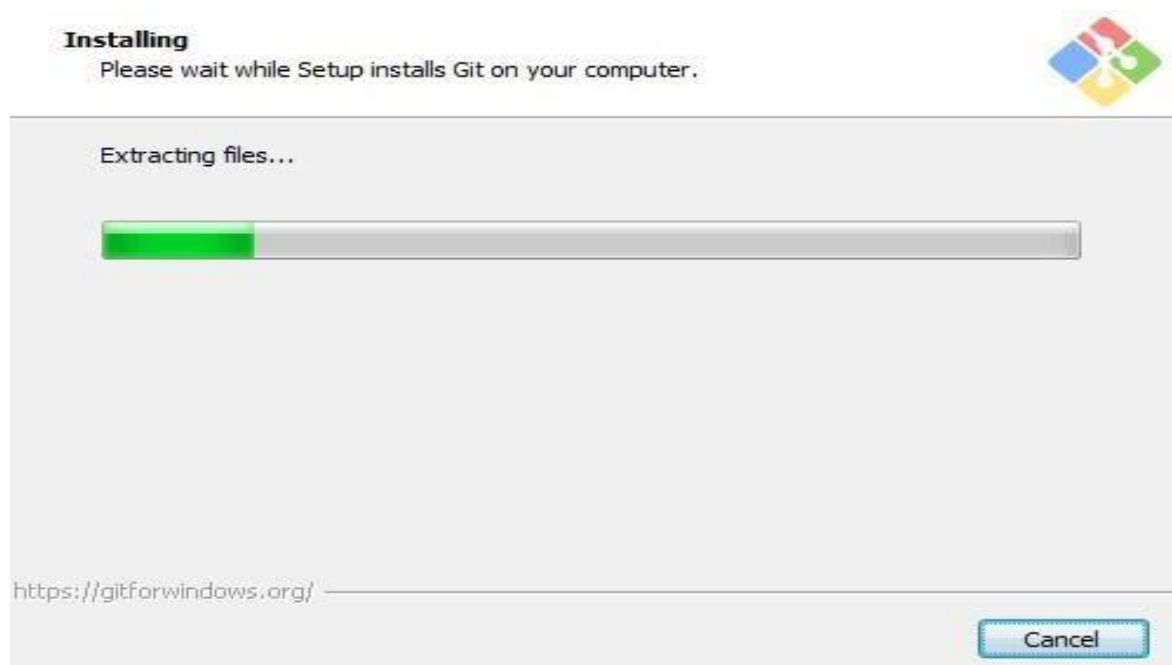
The page looks like as: -



Step 6 =>

The Git is getting download in your system

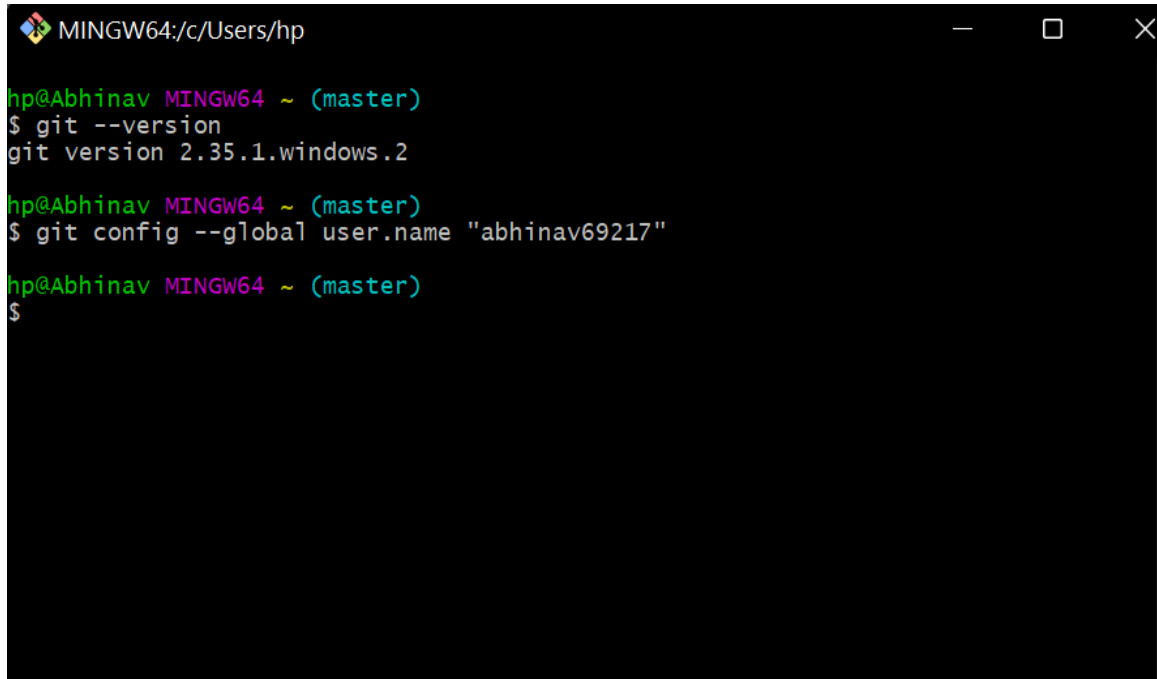
The page looks like as: -



Step 7 =>

You can check that Git is installed by simply type `git --version`

The page looks like as: -

A screenshot of a Windows terminal window titled "MINGW64:/c/Users/hp". The prompt is "hp@Abhinav MINGW64 ~ (master)". The user enters the command "\$ git --version", and the output is "git version 2.35.1.windows.2". The user then enters "\$ git config --global user.name 'abhinav69217'", and the prompt returns. Finally, the user enters "\$", and the prompt returns.

```
MINGW64:/c/Users/hp
hp@Abhinav MINGW64 ~ (master)
$ git --version
git version 2.35.1.windows.2
hp@Abhinav MINGW64 ~ (master)
$ git config --global user.name "abhinav69217"
hp@Abhinav MINGW64 ~ (master)
$
```

GIT is finally installed on your desktop.

4. Setting Up GitHub Account


Step 1 =>

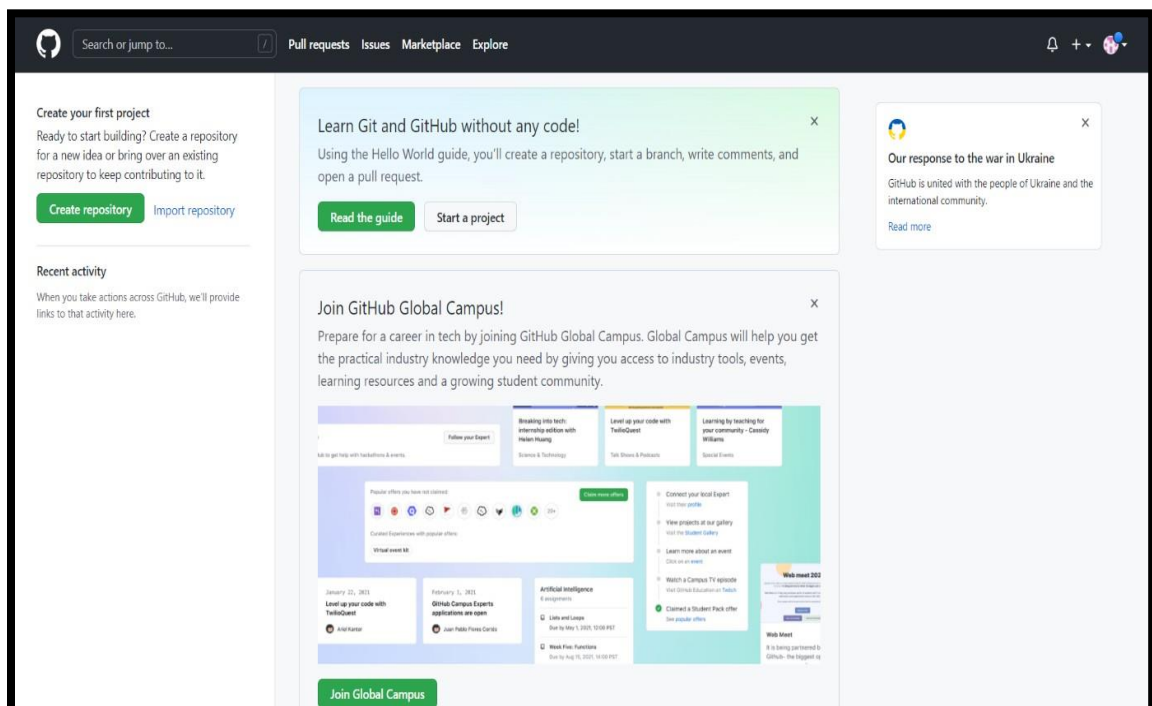
To set up your GitHub account you need to visit <https://github.com/> and click sign-up.

Step 2 =>

Enter your email, username and desired password.



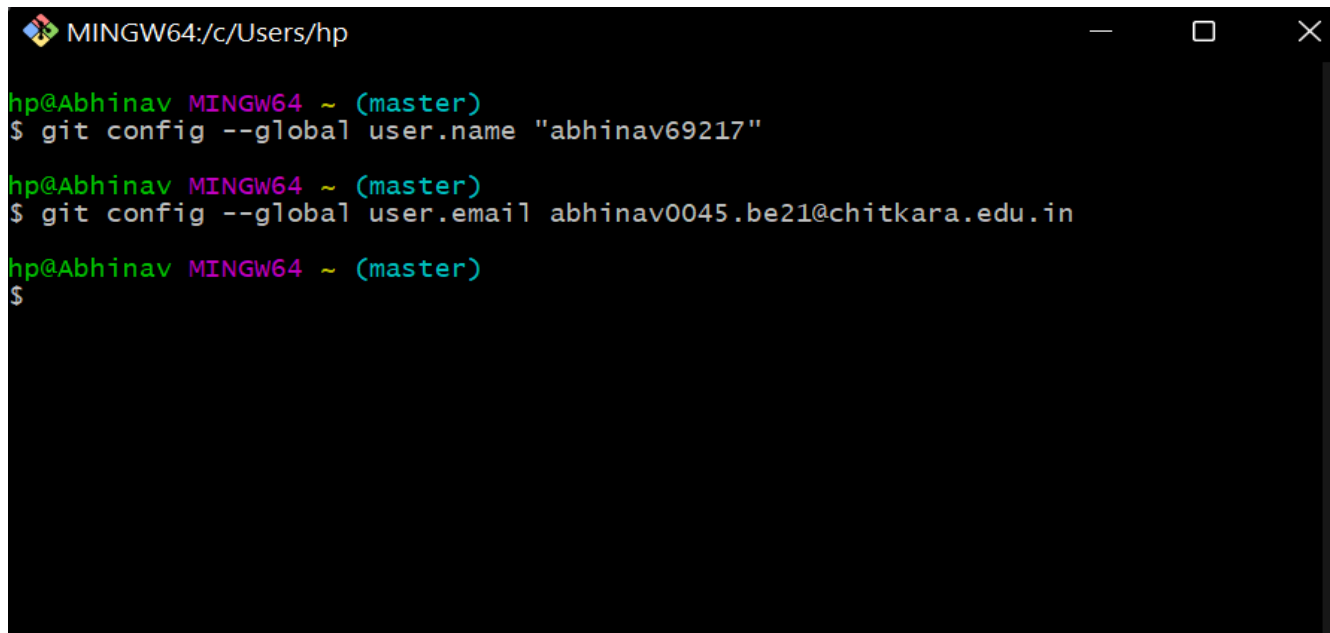
Your account is created 



5. Configuration Of GIT

Step 1 =>

1. config --global user.email "Your Email"
2. Set your username: git config --global user.name "Your Name"



```
MINGW64:/c/Users/hp
hp@Abhinav MINGW64 ~ (master)
$ git config --global user.name "abhinav69217"
hp@Abhinav MINGW64 ~ (master)
$ git config --global user.email abhinav0045.be21@chitkara.edu.in
hp@Abhinav MINGW64 ~ (master)
$
```



Step 2 =>

You can check configuration of Git by typing -

1. git config --list

The page looks like as: -

```

MINGW64:/c/Users/hp
hp@Abhinav MINGW64 ~ (master)
$ git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=E:/Apps/GIT/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
user.name=abhinav69217
user.email=abhinav0045.be21@chitkara.edu.in
core.repositoryformatversion=0
core.filemode=false
core.bare=false
...skipping...
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=E:/Apps/GIT/Git/mingw64/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager-core
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
user.name=abhinav69217
user.email=abhinav0045.be21@chitkara.edu.in
core.repositoryformatversion=0
core.filemode=false
core.bare=false
core.logallrefupdates=true
...skipping...
diff.astextplain.textconv=astextplain

```

6. Program to Generate logs

Advantage of version control systems like git is that it can record changes.

'Git log' command is used to display all these changes that were made by the user in the repository. Log is a record of all the previous commits.

To understand Logs, we need to get familiar with all the commands that are used in making changes to a repository.

- 1) Repository: A repository is a directory that contains all the project-related data.
- 2) Git init: The git init command is used to create a new blank repository.
- 3) Git status: We can list all the untracked, modified and deleted files using the git status command.
- 4) Git add: Adds all the untracked and modified files to the staging area.
- 5) Git commit: Git commit finalizes the changes in the repository. Every commit is saved in the branch you are working in and can be used to revert back to older versions of the project.

Making GIT Repository

Step1:GIT INIT

Initializing a new repository, You Can do it by typing: -

1. git init

The page looks like as: -

```

MINGW64:/d/SCM_1.1
np@Abhinav MINGW64 /d/SCM_1.1
$ git config --global user.name "abhinav69217"

np@Abhinav MINGW64 /d/SCM_1.1
$ git config --global user.email abhinav0045.be21@chitkara.edu.in

np@Abhinav MINGW64 /d/SCM_1.1
$ git init
Initialized empty Git repository in D:/SCM_1.1/.git/

np@Abhinav MINGW64 /d/SCM_1.1 (master)
$ S

```

Step2: ADDING THE FILES TO THE FOLDER

Just like (Samples...)

The page looks like as: -

This PC > New Volume (D:) > SCM_1.1

Name	Date modified	Type	Size
.git	4/9/2022 10:38 AM	File folder	
5_maxElementInArray	3/28/2022 5:57 PM	C++ Source File	1 KB
6_linearSearch	3/28/2022 1:18 PM	C++ Source File	1 KB
7_binarySearch	3/28/2022 5:52 PM	C++ Source File	1 KB

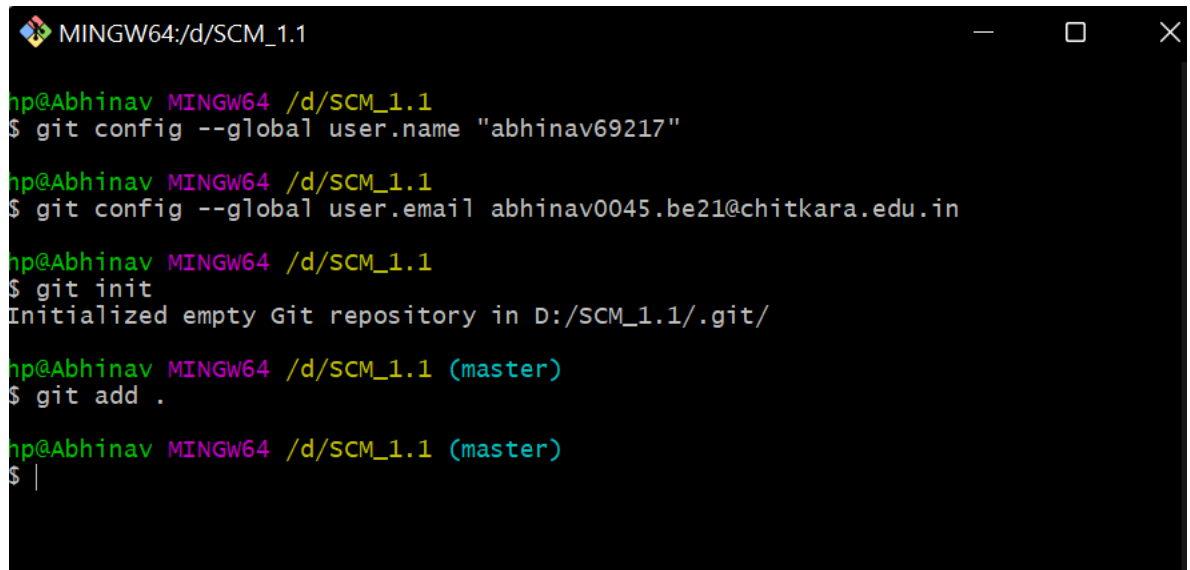
Step3: GIT ADD

The git add command adds a change in the working directory to the staging area.

You Can do it by typing: -

1. `git add .`
2. `git add (current file name)`

The page looks like as: -



```
MINGW64:/d/SCM_1.1

hp@Abhinav MINGW64 /d/SCM_1.1
$ git config --global user.name "abhinav69217"

hp@Abhinav MINGW64 /d/SCM_1.1
$ git config --global user.email abhinav0045.be21@chitkara.edu.in

hp@Abhinav MINGW64 /d/SCM_1.1
$ git init
Initialized empty Git repository in D:/SCM_1.1/.git/

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git add .

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ |
```

Step4:GIT COMMIT

The "commit" command is used to save your changes to the local repository.

You Can do it by typing: -

1. `git commit -m "any text"`

The page looks like as: -


```

MINGW64:/d/SCM_1.1

hp@Abhinav MINGW64 /d/SCM_1.1
$ git config --global user.name "abhinav69217"

hp@Abhinav MINGW64 /d/SCM_1.1
$ git config --global user.email abhinav0045.be21@chitkara.edu.in

hp@Abhinav MINGW64 /d/SCM_1.1
$ git init
Initialized empty Git repository in D:/SCM_1.1/.git/

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git add .

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git commit -m "first commit"
[master (root-commit) e7e416c] first commit
 3 files changed, 75 insertions(+)
 create mode 100644 5_maxElementInArray.cpp
 create mode 100644 6_linearSearch.cpp
 create mode 100644 7_binarySearch.cpp

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$

```

Step5:GIT LOG

Git log will show all the commits made by the author with their time.

After every commit the checksum value (written In yellow color) of the folder changes.

Checksum is used to verify that the data in that file has not been tampered with or manipulated, possibly by a malicious entity.

You Can do it by typing: -

1. git log

The page looks like as: -

```

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git log
commit e7e416c528ace5442b26ee5b5d03b502f0b0a72e (HEAD -> master)
Author: abhinav69217 <abhinav0045.be21@chitkara.edu.in>
Date: Sat Apr 9 10:54:14 2022 +0530

    first commit

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$

```

7. Create and Visualize Branches

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is master.

Step1: CHECKING UP THE BRANCHES

You can check which branch you are working in by using the command

1. 'git branch'

The default branch is always the master branch.
The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git branch
* master
```

Step2: CREATING MULTIPLE BRANCHES

You Can do it by typing: -

1. git branch (BRANCH NAME)

The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git branch abhinav

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git branch
5_maxElementInArray
abhinav
* master

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$
```

Step3: CHANGING BRANCHES

To switch to the other branch
You Can do it by typing: -

1. git checkout (BRANCH NAME)

The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git checkout abhinav
Switched to branch 'abhinav'

hp@Abhinav MINGW64 /d/SCM_1.1 (abhinav)
$
```

Step4: NOW ADD FILE TO THE NEW BRANCH AND COMMIT IT

The page looks like as: -

This PC > New Volume (D:) > SCM_1.1

Name	Date modified	Type
.git	4/9/2022 12:01 PM	File folder
5_maxElementInArray	3/28/2022 5:57 PM	C++ Source File
6_linearSearch	3/28/2022 1:18 PM	C++ Source File
7_binarySearch	3/28/2022 5:52 PM	C++ Source File
helloWorld	4/9/2022 12:01 PM	Text Document

Step5: NOW SWITCH TO BRANCH AND CHECK FILE

The page looks like as: -

```
Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (sample1)
$ git checkout master
Switched to branch 'master'

Nightriders@Inspiron-14-5410 MINGW64 ~/Desktop/Git-Project (master)
$
```

This PC > New Volume (D:) > SCM_1.1

Name	Date modified	Type
.git	4/9/2022 12:01 PM	File folder
5_maxElementInArray	3/28/2022 5:57 PM	C++ Source File
6_linearSearch	3/28/2022 1:18 PM	C++ Source File
7_binarySearch	3/28/2022 5:52 PM	C++ Source File

Now you can see that there is no file named helloWorld.txt in the master branch because we created the file in the sample1 branch. So, it will be exclusive to the feature1 branch.

Step6: GIT MERGING

Now you can merge two branches by command.

1. git merge (BRANCH NAME)

If you want to merge a new branch in master branch you need to first checkout into the master branch and then run the command.

The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (abhinav)
$ git checkout master
Switched to branch 'master'

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git merge
fatal: No remote for the current branch.

hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git merge abhinav
Updating e7e416c..7d7c035
Fast-forward
 helloWorld.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 helloWorld.txt
```

Now you can check the files in the master branch.

This PC > New Volume (D:) > SCM_1.1

Name	Date modified	Type
.git	4/9/2022 12:01 PM	File folder
5_maxElementInArray	3/28/2022 5:57 PM	C++ Source File
6_linearSearch	3/28/2022 1:18 PM	C++ Source File
7_binarySearch	3/28/2022 5:52 PM	C++ Source File
helloWorld	4/9/2022 12:01 PM	Text Document

As you can see the helloworld file is added into the master branch.

Step7: RUNNING GIT LOG

By running git log command on the master branch you can see all the commits made in master as well as the sample1 branch.

The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git log
commit 7d7c03536cb1e0e54cd8662593ba1ffc409681b2 (HEAD -> master, abhinav)
Author: abhinav69217 <abhinav0045.be21@chitkara.edu.in>
Date: Sat Apr 9 12:00:49 2022 +0530

    first commit

commit e7e416c528ace5442b26ee5b5d03b502f0b0a72e (5_maxElementInArray)
Author: abhinav69217 <abhinav0045.be21@chitkara.edu.in>
Date: Sat Apr 9 10:54:14 2022 +0530

    first commit
```

8. Git Lifecycle Description

There are three stages for git lifecycle:

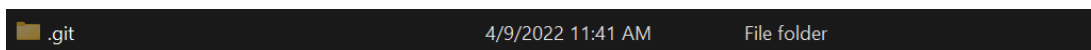
- 1) Working directory
- 2) Staging area
- 3) Git repository

Working Directory:

The working directory is the folder in your local computer where the project files and folders are stored.

The local directory is created by the command 'git init' which creates a '.git' named folder which is used to track the files in the directory.

'git folder' is generally hidden but can be tracked enabling hidden files.



Staging area:

The staging area has those files which are supposed to go to the next commit. Only those files which are needed to go to the next commit stay in the staging area.

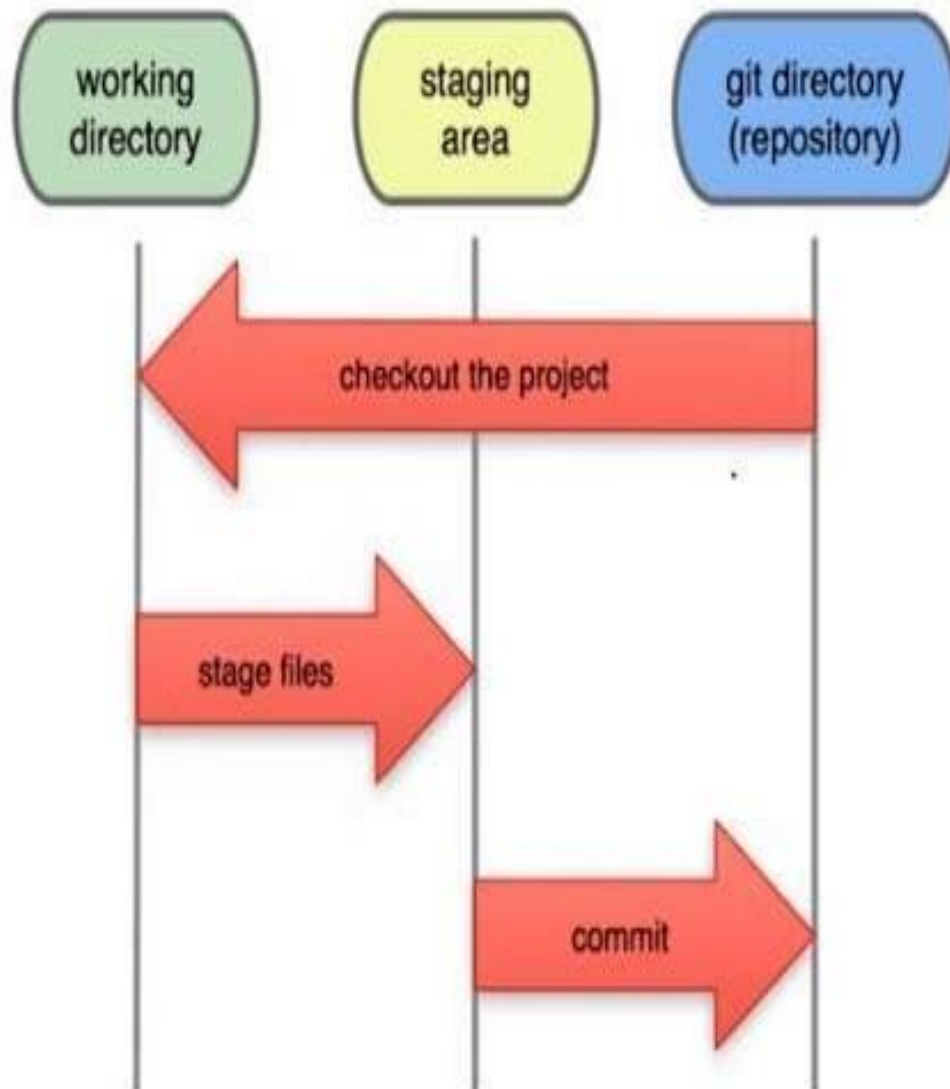
You can shift the files to the git repository by using the command 'git add --a'.

Git repository:

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about.

You can commit files by using command 'git commit -m "message"'

Local Operations



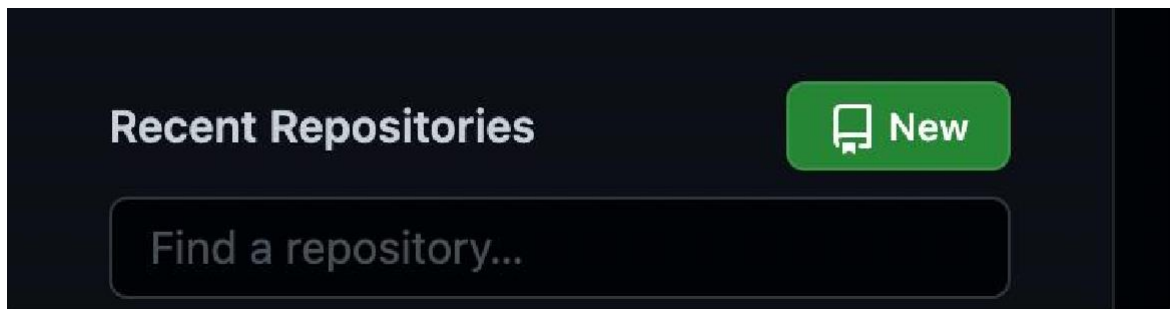
9. Uploading Data on GitHub

NOTE-

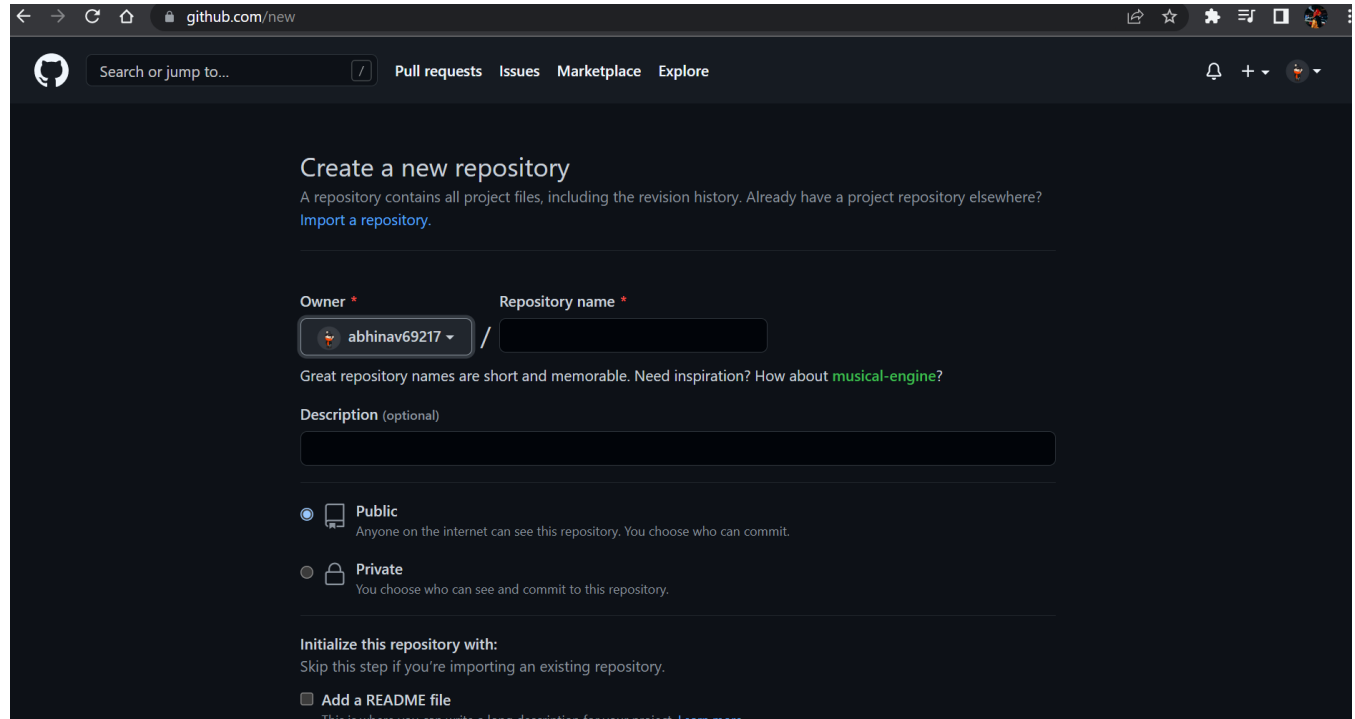
YOU HAVE TO MAKE A REPOSITORY IN GITHUB.

Step1) CREATING REPOSITORY IN GITHUB

The page looks like as: -



By clicking on new you are able to make a new repository.



Write the repository name and click on next.



Your GITHUB Repository has been created.

Step2) GIT ADDING REMOTE BRANCH

Git stores a branch as a reference to a commit, and a branch represents the tip of a series of commits.

You Can do it by typing: -

1. `git branch -M main`

The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (master)
$ git branch -M main
```

Step3) GIT ADDING REMOTE ORIGIN

Is a Git repository that's hosted on the Internet

You Can do it by typing: -

1. `git remote add origin (URL)`

The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (main)
$ git remote add origin https://github.com/abhinav69217/SCM_1.1.git
```

Step4) GIT PUSHING

The git push command is used to upload local repository content to a remote repository.

You Can do it by typing: -

1. `git push -u origin main`

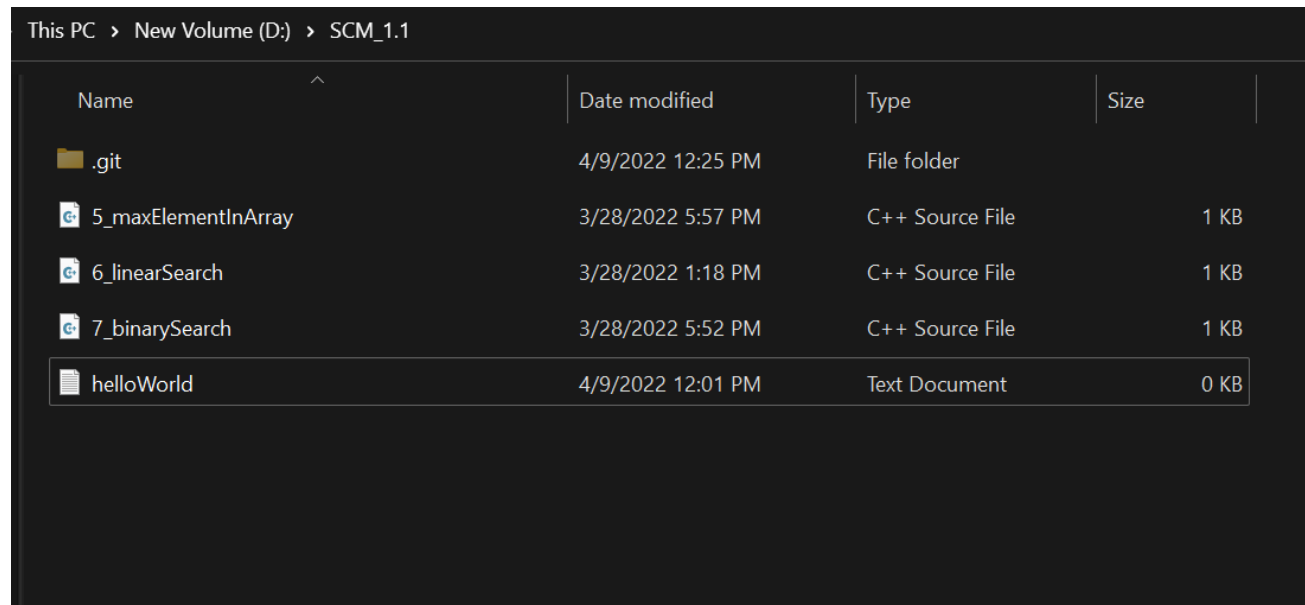
The page looks like as: -

```
hp@Abhinav MINGW64 /d/SCM_1.1 (main)
$ git push -u origin main
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.17 KiB | 1.17 MiB
Total 8 (delta 1), reused 0 (delta 0), pack-reus
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/abhinav69217/SCM_1.1.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Final Result

1. Document in your system

The page looks like as: -



2. Document in your GITHUB Repository

The page looks like as: -

