Experiment No. – 2				
Date of Performance:	18/7/24			
Date of Submission:	25/7/24			
Program Execution/ formation/ correction/ ethical practices (06)	Timely Submission (01)	Viva (03)	Experiment Total (10)	Sign with Date

Experiment No. 2

2.1 Aim:To understand Version Control System by installing git and create GitHub Account.

2.2 Course Outcome: CO1

2.3 Learning Objectives: Remember the importance of DevOps tools used in software development life cycle.

2.4 Requirement:

1. Software Requirements:
    - A computer with an internet connection.
    - An operating system (Windows, macOS, or Linux).
2. Accounts:
    - A valid email address to create a GitHub account.

2.5 Related Theory:

What is a Version Control System (VCS)?

- Definition: A VCS is a tool that helps developers manage changes to source code over time. It allows multiple people to work on a project simultaneously without conflicts.
- Types:
    - Centralized VCS: All files are stored in a central server (e.g., Subversion).
    - Distributed VCS: Each user has a complete copy of the repository (e.g., Git).

What is Git?

- Definition: Git is a distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
- Key Features:
    - Branching and Merging: Allows for multiple lines of development.
    - History Tracking: Keeps a detailed history of changes.
    - Collaboration: Facilitates teamwork by managing contributions from multiple users.

What is GitHub?

- Definition: GitHub is a cloud-based platform that uses Git for version control and allows developers to host, share, and collaborate on projects.
- Key Features:
    - Repositories: Store project files and version history.
    - Pull Requests: Facilitate code review and collaboration.
    - Issues: Track bugs and feature requests.

2.6 Procedure:

Step 1: Install Git

1. Download Git:
    - Go to the official Git website: git-scm.com
    - Choose the appropriate version for your operating system.
2. Install Git:
    - Run the installer and follow the setup instructions.
    - For Windows, select default options unless specific needs arise.
    - For macOS, you might be prompted to install Xcode command line tools.
3. Verify Installation:
    - Open a terminal (Command Prompt, Terminal, etc.).
    - Type git --version to check if Git is installed correctly.

Step 2: Configure Git

Set Your Name and Email:

git config --global user.name "Your Name"

git config --global user.email "youremail@example.com"

1. Check Configuration:
   git config --list

Step 3: Create a GitHub Account

1. Sign Up:
    - Go to github.com.
    - Click on "Sign up" and fill in the required information (username, email, password).
2. Verify Email:
    - Check your email for a verification link from GitHub and click to confirm.
3. Create a Repository:
    - After logging in, click on the "+" icon in the upper right corner and select "New repository".
    - Name your repository, add a description, and decide whether it should be public or private.
    - Click "Create repository".

Step 4: Use Git and GitHub Together

1. Clone a Repository:
    - In GitHub, go to your repository and copy the clone URL.

In your terminal, run:
git clone <repository-url>

2. Make Changes and Commit:
   Navigate into the cloned repository folder:
   cd repository-name

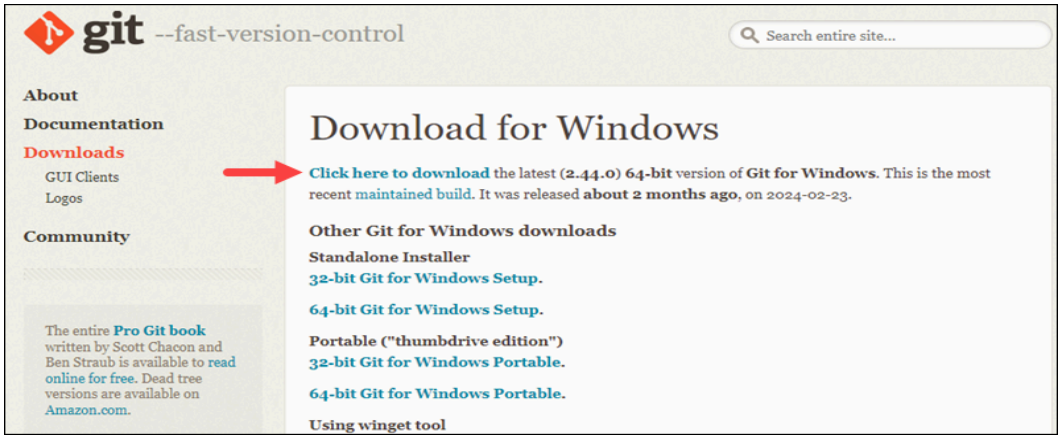
Create or edit files, then stage and commit:
git add .
git commit -m "Initial commit"

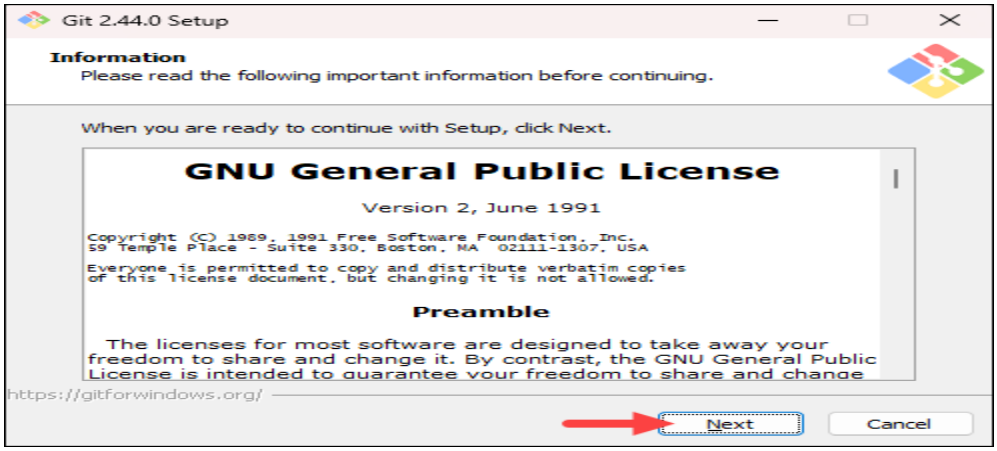
3. Push Changes to GitHub:
   git push origin main

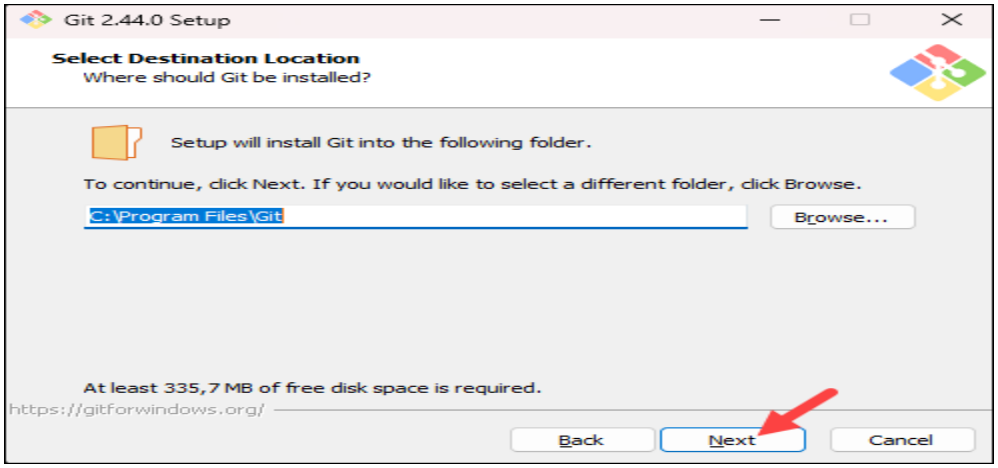
2.7 Program and Output:

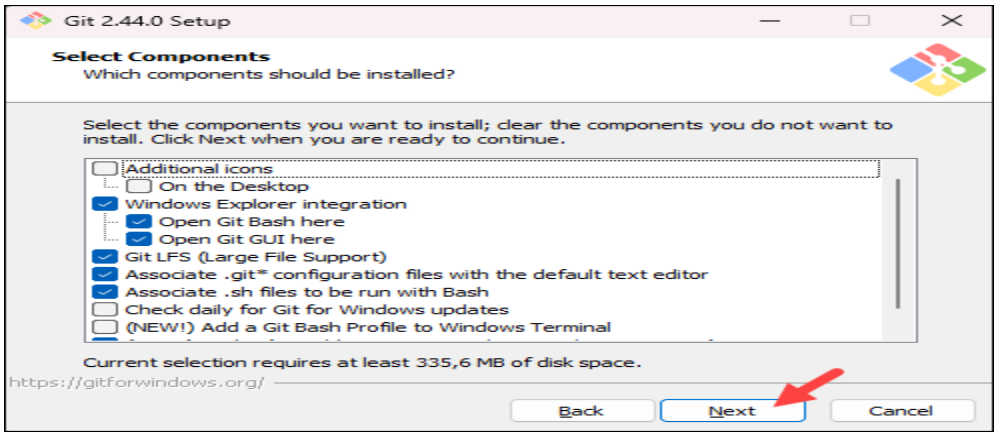
1. Installing Git

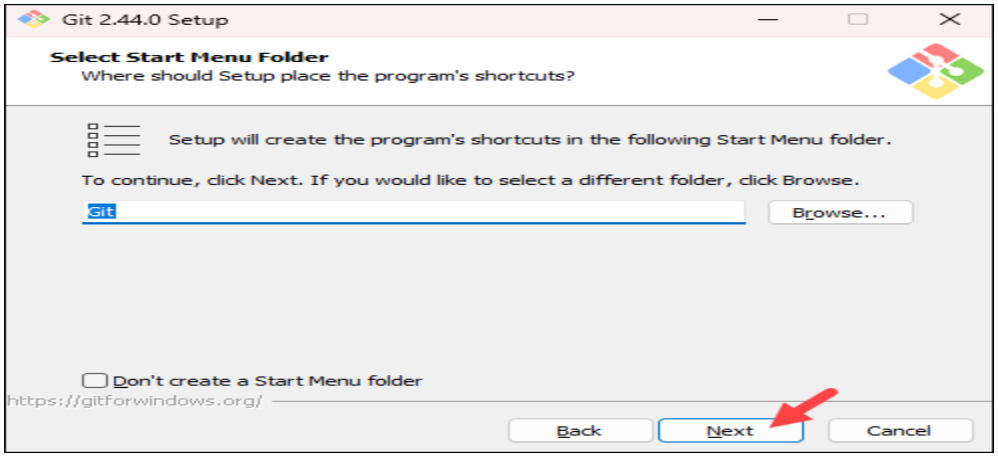
Step for installing git:

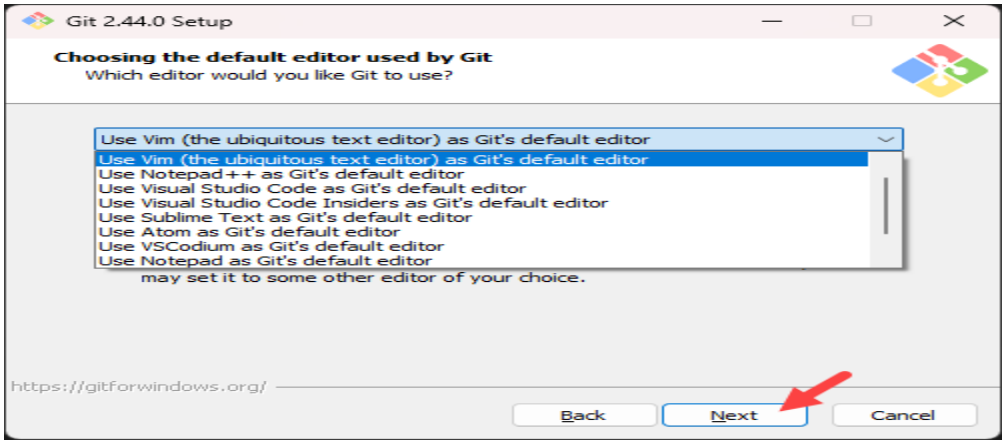


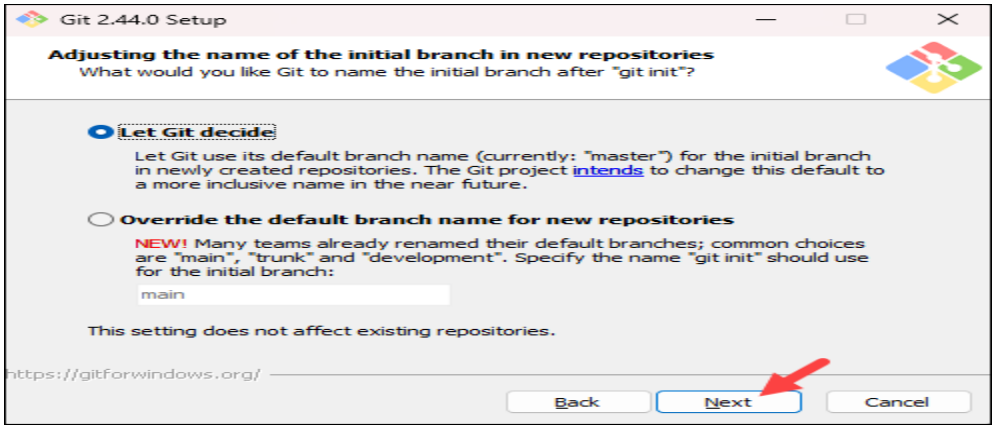


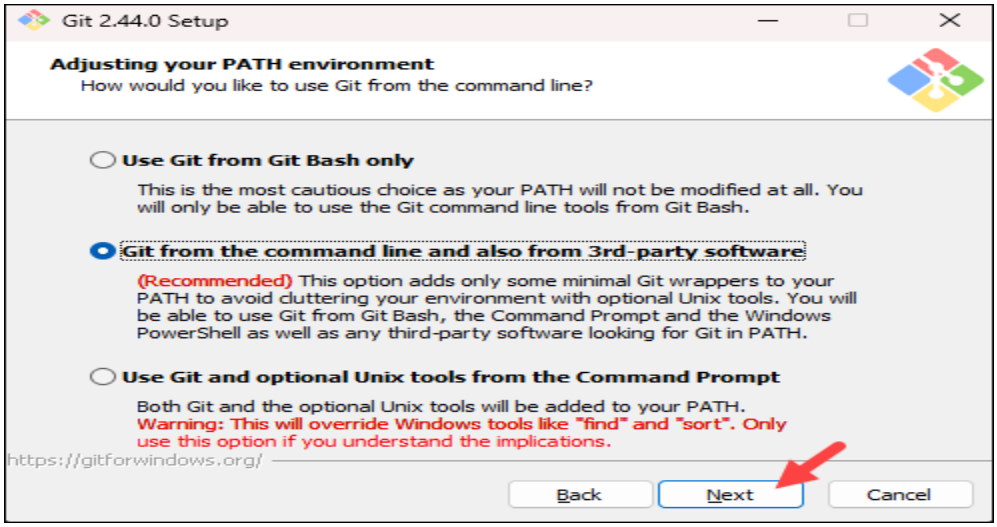

Git 2.44.0 Setup
Select Destination Location
Where should Git be installed?
Setup will install Git into the following folder.
To continue, click Next. If you would like to select a different folder, click Browse.
C:\Program Files\Git
Browse...
At least 335,7 MB of free disk space is required.
https://gitforwindows.org/
Back
Next
Cancel

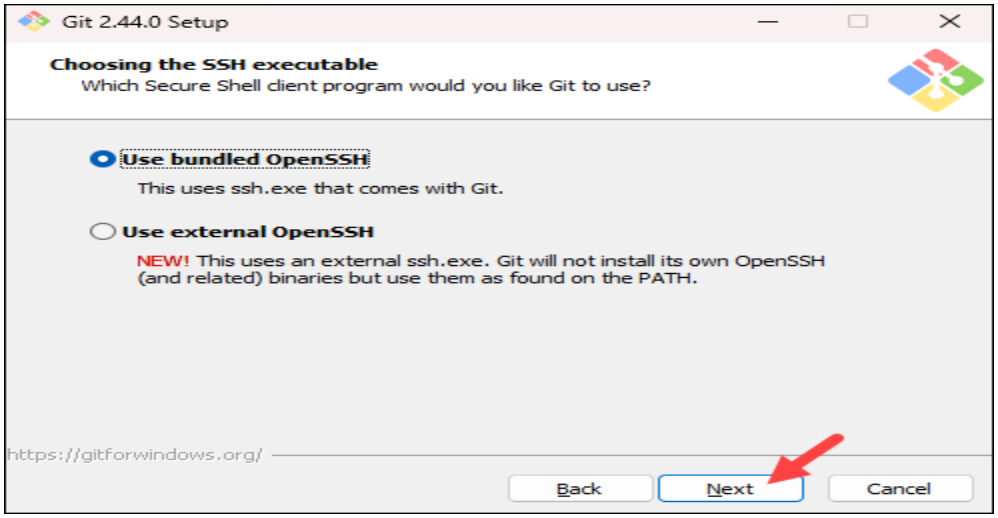

Git 2.44.0 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Open Git Bash here
Open Git GUI here
Git LFS (Large File Support)
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Check daily for Git for Windows updates
(NEW!) Add a Git Bash Profile to Windows Terminal
Current selection requires at least 335,6 MB of disk space.
https://gitforwindows.org/
Back
Next
Cancel

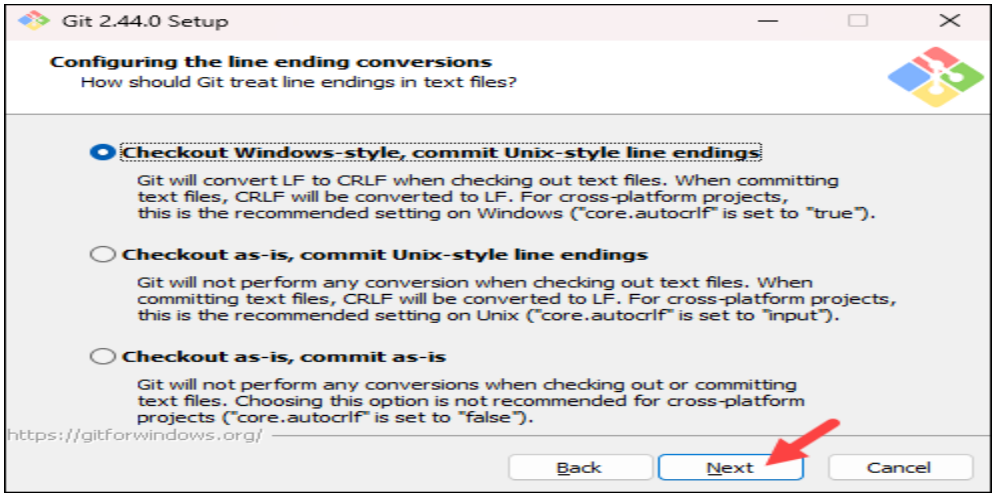

Git 2.44.0 Setup
Select Start Menu Folder
Where should Setup place the program's shortcuts?
Setup will create the program's shortcuts in the following Start Menu folder.
To continue, click Next. If you would like to select a different folder, click Browse.
Git
Browse...
Don't create a Start Menu folder
https://gitforwindows.org/
Back
Next
Cancel

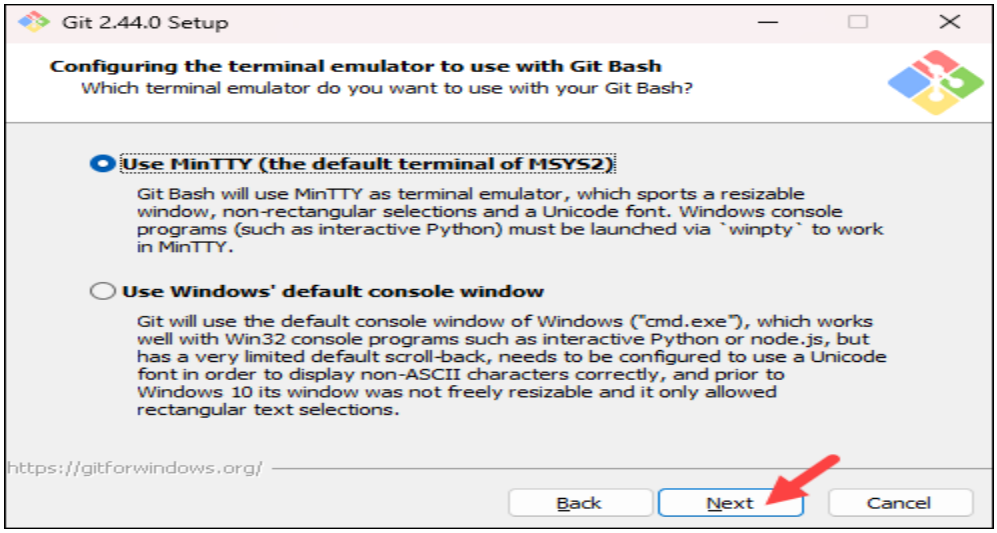

Git 2.44.0 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
Use Vim (the ubiquitous text editor) as Git's default editor
Use Notepad++ as Git's default editor
Use Visual Studio Code as Git's default editor
Use Visual Studio Code Insiders as Git's default editor
Use Sublime Text as Git's default editor
Use Atom as Git's default editor
Use VSCodium as Git's default editor
Use Notepad as Git's default editor
may set it to some other editor of your choice.
https://gitforwindows.org/
Back
Next
Cancel

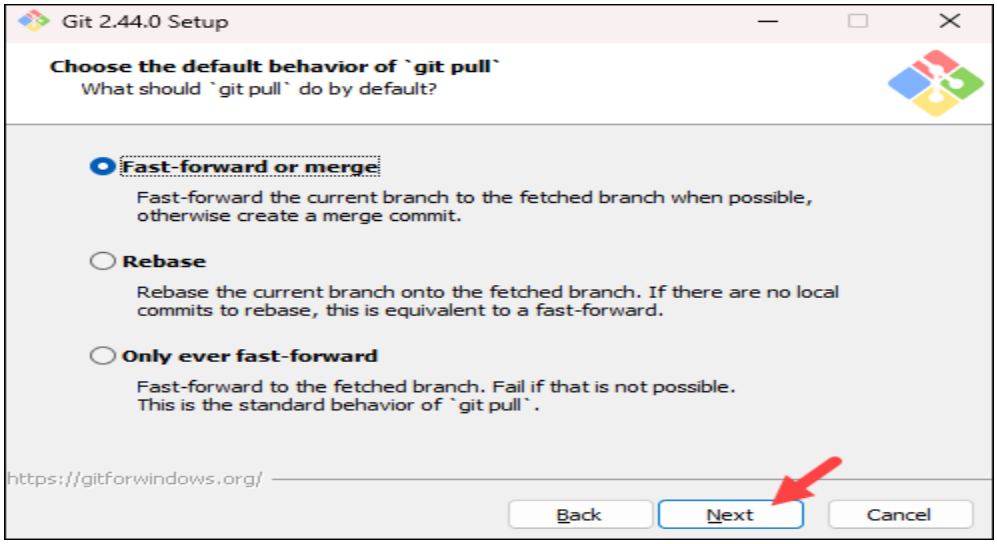

Git 2.44.0 Setup
Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?
Let Git decide
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project intends to change this default to a more inclusive name in the near future.
Override the default branch name for new repositories
NEW! Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:
main
This setting does not affect existing repositories.
https://gitforwindows.org/
Back
Next
Cancel

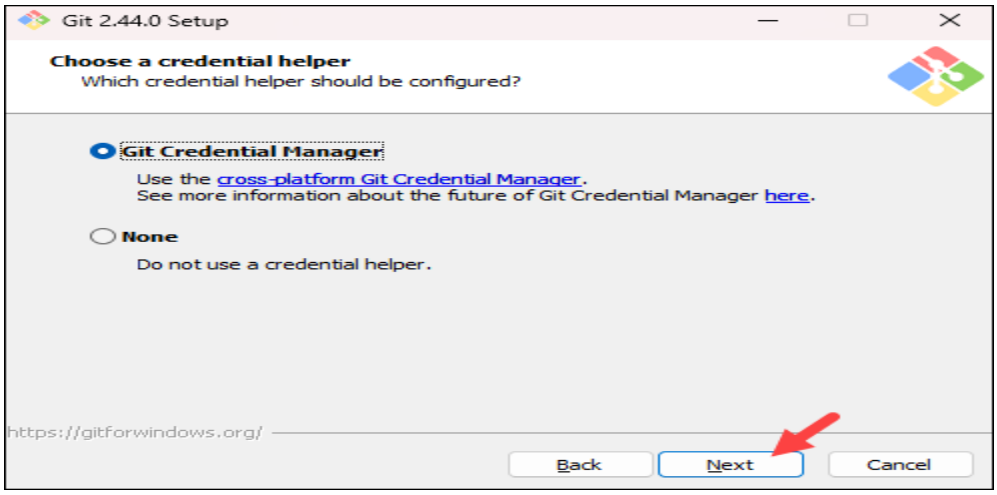

Git 2.44.0 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line 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

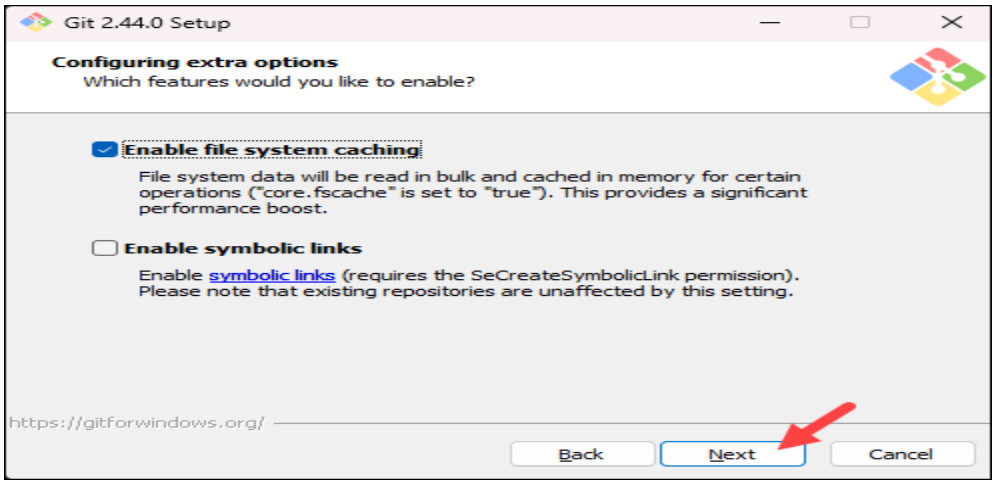

Git 2.44.0 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Back
Next
Cancel

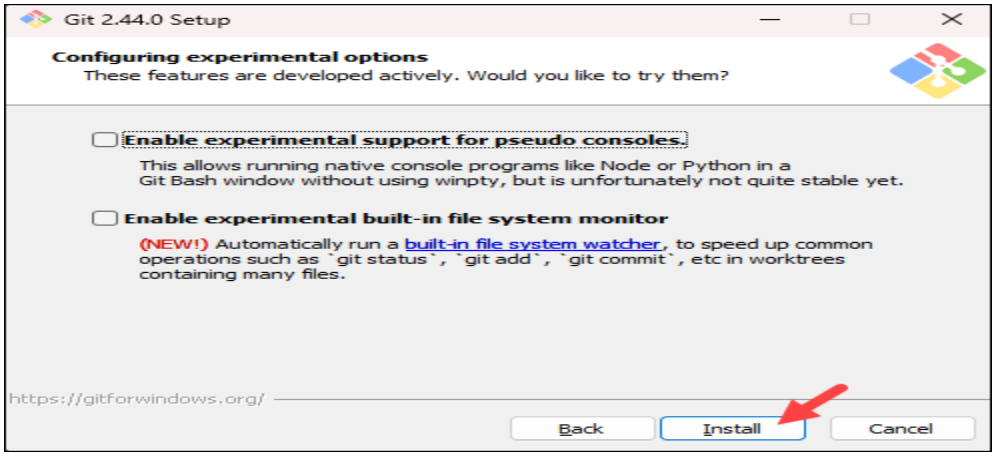

Git 2.44.0 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
https://gitforwindows.org/
Back
Next
Cancel

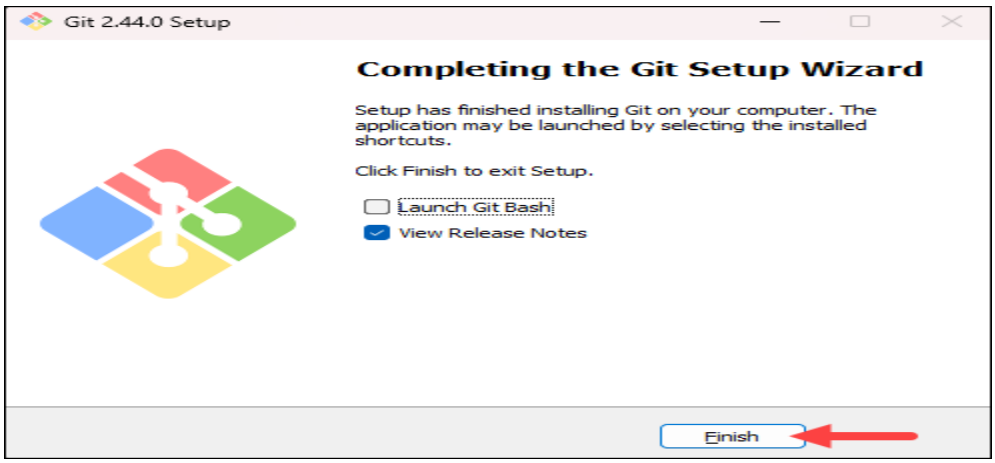

Git 2.44.0 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works well with Win32 console programs such as interactive Python or node.js, but has a very limited default scroll-back, needs to be configured to use a Unicode font in order to display non-ASCII characters correctly, and prior to Windows 10 its window was not freely resizable and it only allowed rectangular text selections.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.44.0 Setup
Choose the default behavior of `git pull`
What should `git pull` do by default?
Fast-forward or merge
Fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.
Rebase
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.
Only ever fast-forward
Fast-forward to the fetched branch. Fail if that is not possible. This is the standard behavior of `git pull`.
https://gitforwindows.org/
Back
Next
Cancel

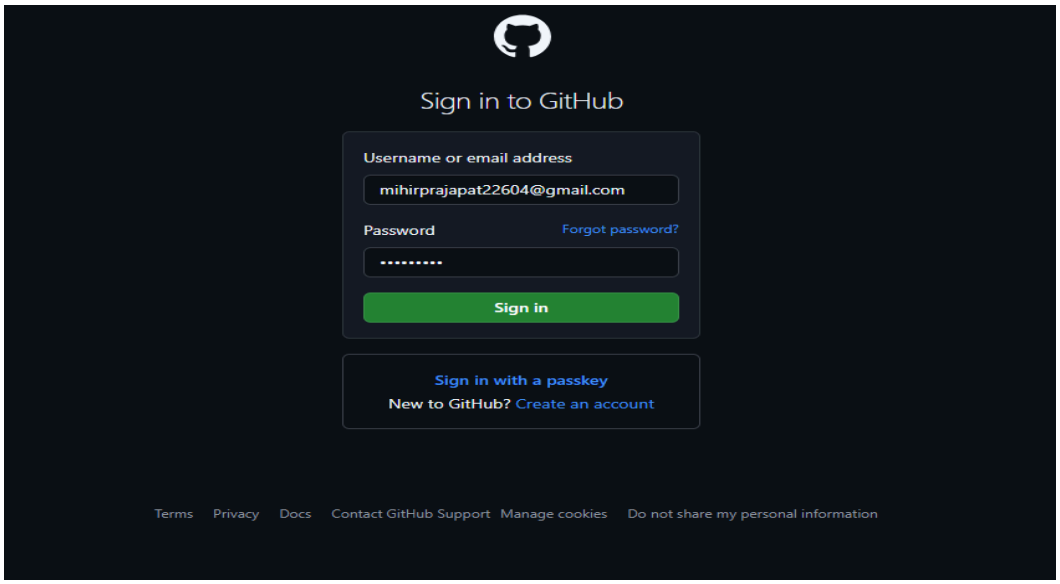

Git 2.44.0 Setup
Choose a credential helper
Which credential helper should be configured?
Git Credential Manager
Use the cross-platform Git Credential Manager.
See more information about the future of Git Credential Manager here.
None
Do not use a credential helper.
https://gitforwindows.org/
Back
Next
Cancel


Git 2.44.0 Setup
Configuring extra options
Which features would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost.
Enable symbolic links
Enable symbolic links (requires the SeCreateSymbolicLink permission).
Please note that existing repositories are unaffected by this setting.
https://gitforwindows.org/
Back
Next
Cancel





3. Github installation:



Welcome to GitHub!
Let's begin the adventure

Enter your email*
✓ lamiyij181@rinseart.com

Create a password*
✓ Lama@181

Enter a username*
→ MiHir37

Continue

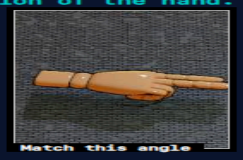
Username MiHir37 is not available.
MiHir37-ui, MiHir37-source, or MiHir37639 are available.

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

Welcome to GitHub!
Let's begin the adventure

Verify your account

Use the arrows to rotate the object to face in the direction of the hand. (1 of 2)

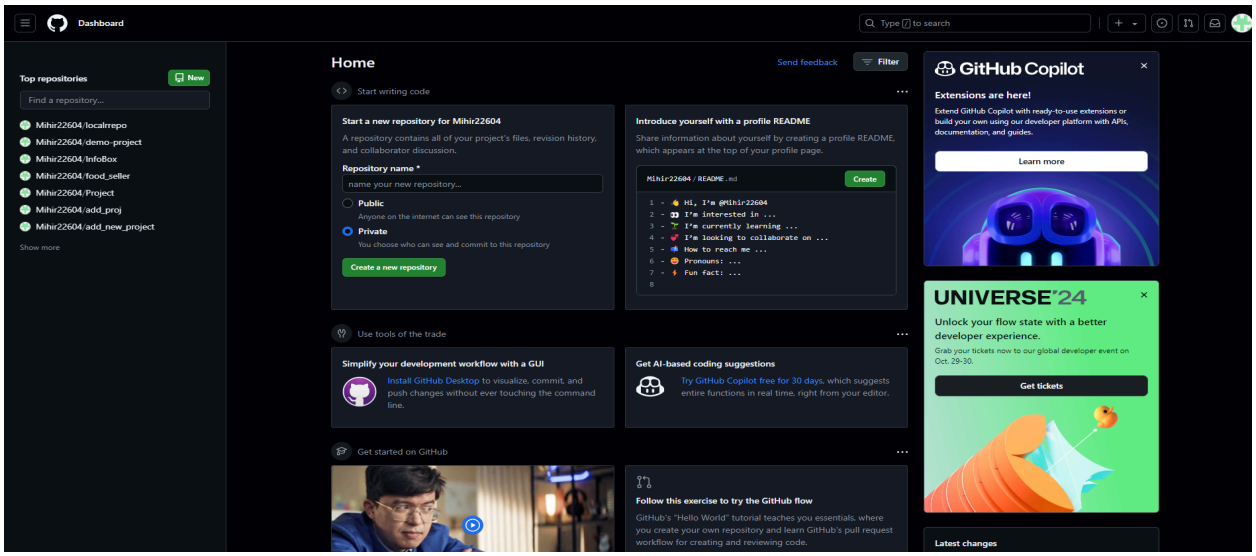


Submit

Audio

Restart

By creating an account, you agree to the Terms of Service. For more information about GitHub's privacy practices, see the GitHub Privacy Statement. We'll occasionally send you account-related emails.

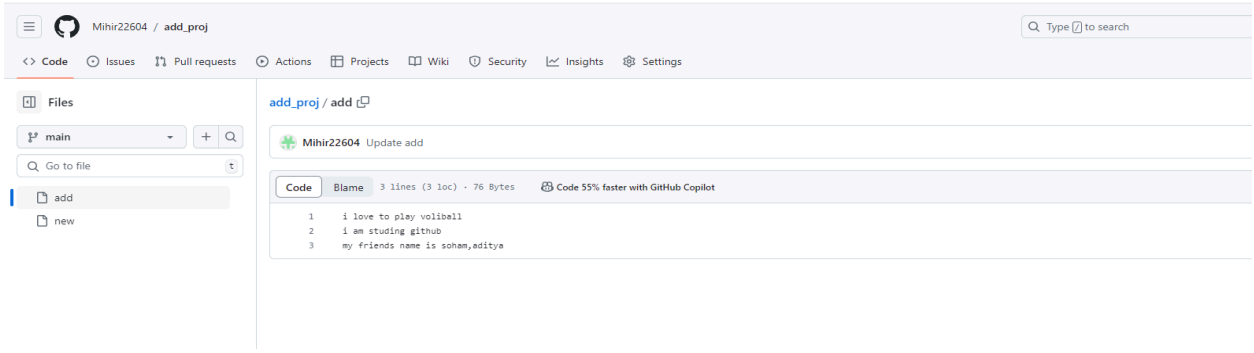


```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project/add_proj (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 368 bytes | 368.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Mihir22604/add_proj.git
   4878a92..a88697b  main -> main

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project/add_proj (main)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 976 bytes | 97.00 KiB/s, done.
From https://github.com/Mihir22604/add_proj
 * branch            main       -> FETCH_HEAD
   a88697b..17e25f2  main       -> origin/main
Updating a88697b..17e25f2
Fast-forward
 add | 1 +
 1 file changed, 1 insertion(+)

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project/add_proj (main)
$ cat add
i love to play voliball
i am studing github
my friends name is soham,aditya

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project/add_proj (main)
$ history
    1  git --version
    2  mkdir arkam
    3  cd arkam
    4  git init
    5  git config --global
    6  git config --global user.name"arkam"
    7  git config --global user.email "arkam.169380@sakec.ac.in"
    8  git config --global --list
    9  git config --global user.name "arkam"
   10  git config --global user.email "arkam.169380@sakec.ac.in"
   11  git config --global --list
   12  git status
   13  git commit -m "firstcommit"
   14  git add 'file1'
   15  git add 'arkam'
   16  git add.
   17  git add .
   18  git commit -am "file1"
   19  git add "file1"
   20  git remote origin https://github.com/arkamsunesara/arkam.git
   21  git --version
   22  mkdir file1
   23  cd file1
   24  touch f1
   25  vim f1
   26  git status
   27  git init
   28  git status
   29  git add .
   30  git commit -m "basic info"
   31  git status
   32  vim f1
   33  clear
   34  mkdir git-dvcs
```




```
lab503@SAKEC-LAB503C5 MINGW64 ~
$ mkdir new

lab503@SAKEC-LAB503C5 MINGW64 ~
$ cd new
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new
$ git config --global
usage: git config [<options>]

Config file location
    --global              use global config file
    --system              use system config file
    --local               use repository config file
    --worktree            use per-worktree config file
    -f, --file <file>     use given config file
    --blob <blob-id>      read config from given blob object

Action
    --get                 get value: name [value-pattern]
    --get-all             get all values: key [value-pattern]
    --get-regexp          get values for regexp: name-regex [value-pattern]
    --get-urlmatch        get value specific for the URL: section[.var] URL
    --replace-all         replace all matching variables: name value [value-pattern]
    --add                 add a new variable: name value
    --unset               remove a variable: name [value-pattern]
    --unset-all           remove all matches: name [value-pattern]
    --rename-section      rename section: old-name new-name
    --remove-section      remove a section: name
    -l, --list            list all
    --fixed-value         use string equality when comparing values to 'value-pattern'
    -e, --edit            open an editor
    --get-color           find the color configured: slot [default]
    --get-colorbool       find the color setting: slot [stdout-is-tty]

Type
    -t, --type <type>     value is given this type
    --bool                value is "true" or "false"
    --int                 value is decimal number
    --bool-or-int         value is --bool or --int
    --bool-or-str         value is --bool or string
    --path                value is a path (file or directory name)
    --expiry-date         value is an expiry date

Other
    -z, --null            terminate values with NUL byte
    --name-only           show variable names only
    --includes            respect include directives on lookup
    --show-origin         show origin of config (file, standard input, blob, command line)
    --show-scope          show scope of config (worktree, local, global, system, command)
    --default <value>     with --get, use default value when missing entry
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new
$ git config --global user.name "Mihir"

lab503@SAKEC-LAB503C5 MINGW64 ~/new
$ git config --global user.email "mihirprajapat22604@gmail.com"

lab503@SAKEC-LAB503C5 MINGW64 ~/new
$ git config --global --list
user.email=mihirprajapat22604@gmail.com
user.name=Mihir
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new
$ mkdir git-demo-project

lab503@SAKEC-LAB503C5 MINGW64 ~/new
$ cd git-demo-project

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project
$ git init
Initialized empty Git repository in C:/Users/LAB503/new/git-demo-project/.git/

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ ls -a
./  ../  .git/

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ rm -rf .git/

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project
$ ls -al
total 0
drwxr-xr-x 1 lab503 1049089 0 Jul 18 13:41 ./
drwxr-xr-x 1 lab503 1049089 0 Jul 18 13:37 ../
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ touch add

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ vim add

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git add .
warning: in the working copy of 'add', LF will be replaced by CRLF the next time Git touches it

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git add "add"

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git status
On branch master

No commits yet

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

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ touch index.html

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git commit -m "first commit"
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        index.html

nothing added to commit but untracked files present (use "git add" to track)

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git add .

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git commit -am "expree commit"
[master aae1aa6] expree commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ vim index.html

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git add .
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git add "index.html"

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git commit -m "new add"
[master 3f46c05] new add
 1 file changed, 2 insertions(+)
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git checkout -- index.html
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ nano index.html
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git log
commit 3f46c05220dc117d5792e2615868e838c651c4c8 (HEAD -> master)
Author: Mihir <mihirprajapat22604@gmail.com>
Date:   Thu Jul 18 13:57:36 2024 +0530

    new add

commit aae1aa620f8992d4144efe3b7a07158d95bc0309
Author: Mihir <mihirprajapat22604@gmail.com>
Date:   Thu Jul 18 13:54:10 2024 +0530

    expree commit

commit c331a817b6e39cfbcd7b984cd8e8bb4b272e19bb
Author: Mihir <mihirprajapat22604@gmail.com>
Date:   Thu Jul 18 13:47:07 2024 +0530

    Todays info
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git log --oneline
3f46c05 (HEAD -> master) new add
aae1aa6 expree commit
c331a81 Todays info
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git log --oneline add
c331a81 Todays info
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git log --oneline  aae1aa620f8992d4144efe3b7a07158d95bc0309
aae1aa6 expree commit
c331a81 Todays info
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git log --oneline -n 2
3f46c05 (HEAD -> master) new add
aae1aa6 expree commit
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git branch
  apple
* master

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git checkout apple
Switched to branch 'apple'

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (apple)
$ git branch
* apple
  master

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (apple)
$ vim apple

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (apple)
$ git status
On branch apple
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        apple

nothing added to commit but untracked files present (use "git add" to track)

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (apple)
$ git add .
warning: in the working copy of 'apple', LF will be replaced by CRLF the next time Git touches it

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (apple)
$ git add "apple"

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (apple)
$ git commit -m "change in branch"
[apple 172839f] change in branch
 1 file changed, 1 insertion(+)
 create mode 100644 apple

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (apple)
$ git checkout master
Switched to branch 'master'

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git merge apple
Updating 3f46c05..172839f
Fast-forward
 apple | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 apple

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git status
On branch master
nothing to commit, working tree clean

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ vim master

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ ls
add  apple  index.html  master
```

```
lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (master)
$ git branch -M main

lab503@SAKEC-LAB503C5 MINGW64 ~/new/git-demo-project (main)
$ git branch
  apple
* main
```

2.8 Conclusion: You have now successfully installed Git, configured it, created a GitHub account, and linked your local repository to GitHub. Understanding version control with Git and utilizing GitHub for collaboration is an invaluable skill for modern software development. With these tools, you can effectively manage your projects, track changes, and collaborate with others, laying the groundwork for successful development practices. As you continue to explore Git and GitHub, consider diving deeper into branching strategies, pull requests, and collaborative workflows.

2.9 Questions:

1] What is a Version Control System, and why is it important in software development?

Ans: A Version Control System (VCS) is a tool that helps developers track and manage changes to code over time. It allows multiple users to collaborate on projects, keeps a history of changes, and enables reverting to previous versions if needed.

Importance in Software Development:

2. Collaboration: Multiple developers can work on the same project without conflicts.
3. History Tracking: Keeps a detailed record of changes, facilitating accountability and transparency.
4. Branching and Merging: Supports experimentation and parallel development through branches.
5. Backup and Recovery: Protects against data loss by maintaining backups of every version of the code.

2] What is branching in Git, and why is it useful?

Ans:- Branching in Git is a feature that allows developers to create independent lines of development within a repository. Each branch can contain its own changes and can be worked on simultaneously without affecting the main codebase.

Why It's Useful:

1. Isolation: Changes can be developed and tested independently without impacting the main branch (usually main or master).
2. Experimentation: Developers can try new features or fixes in branches without the risk of breaking existing functionality.
3. Collaboration: Teams can work on different features or bug fixes concurrently and merge them back into the main codebase when ready.
4. Version Control: Facilitates easier tracking of changes related to specific features or issues, improving organization and clarity.