St. Francis Institute of Technology, Mumbai-400 103

Department Of Information Technology

A.Y. 2024-2025

Class: TE-ITA/B, Semester: V

Subject: **DevOps Lab**

Experiment – 2: To understand version control system/ source code management, install git and create a GitHub account

1. **Aim:** To understand version control using Git and create a GitHub account
2. **Objectives:** Aim of this experiment is that, the students will be able
   - To be aware of different Version Control tools like GIT and GitHub
   - To obtain complete knowledge of the "version control system" to effectively track changes augmented with Git and GitHub
3. **Outcomes:** After study of this experiment, the students will be able to
   - GIT Installation
   - Version Control
   - Working with remote repository
4. **Prerequisite:** Knowledge of software engineering concept of version control
5. **Requirements:** Git, Personal Computer, Windows operating system, browser, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**
   Brief Theory: Refer shared material
7. **Laboratory Exercise**
   A. **Procedure:**
      a. **Answer the following:**
         - Explain version control and its types
         - What is Git and GitHub?
         - Explain different Git commands

      b. **Execute following on Git and GitHub (Refer the shared material) and attach screenshots:**
         - Git installation
         - Git commands
         - GitHub account creation
8. **Post-Experiments Exercise**
   A. **Extended Theory:**
      Nil
   B. **Questions:**
      - What are the different Git areas? Explain with diagram
      - What is a Git conflict?
   C. **Conclusion:**
      - Write what was performed in the experiment.
      - Write the significance of the topic studied in the experiment.
   D. **References:**
      https://github.com/
      https://guides.github.com/activities/hello-world/
      https://git-scm.com/docs/gittutorial

Name: Vishal Rajesh Mahajan Devops EXP 2
Class: TE IT A Roll No: 62

7. Laboratory Exercise:

A. Answer the following questions

1. Explain Version Control and Its Types
Answer: Version Control is a system that tracks and manages changes to files over time, allowing multiple collaborators to work together. The main types are:

- Local Version Control: Keeps track of changes on a local machine.
- Centralized Version Control (CVCS): Uses a single server to store all versions of files; examples include SVN.
- Distributed Version Control (DVCS): Each user has a full copy of the repository; examples include Git and Mercurial.

2. **What is Git and GitHub?**
Answer: Git is a distributed version control system that tracks changes in source code during software development, allowing multiple developers to work together.

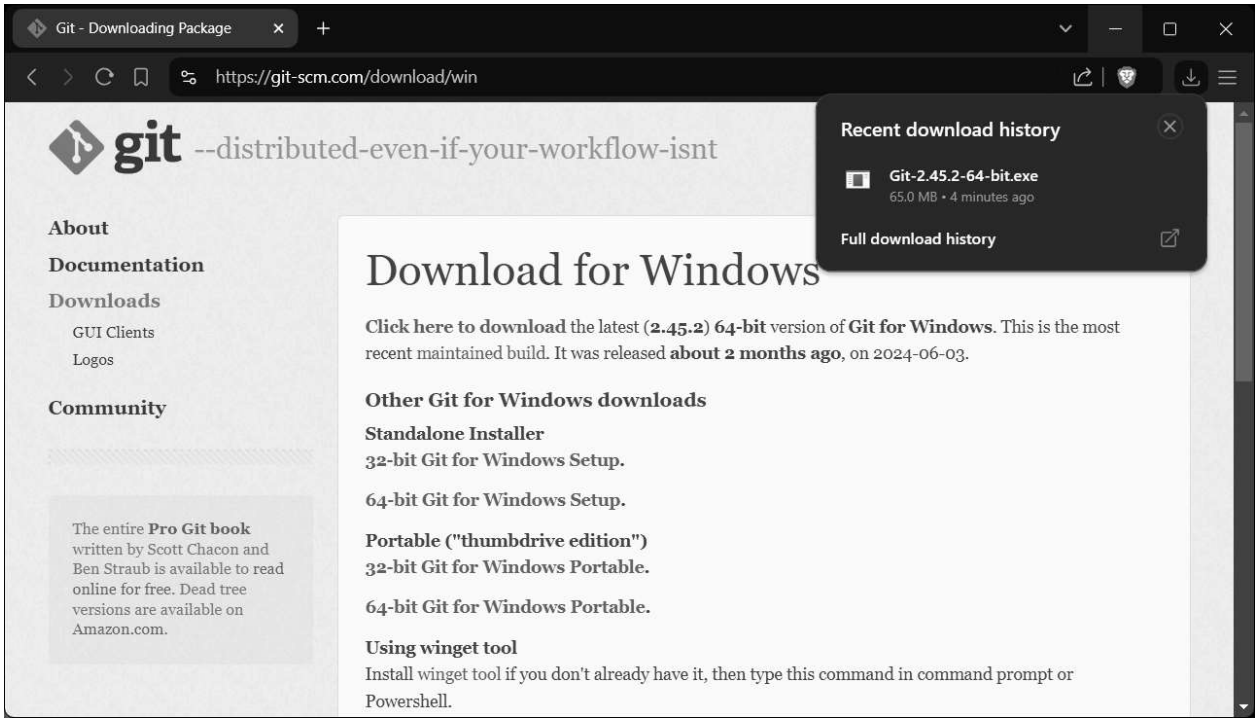
GitHub is an online platform that hosts Git repositories, offering features like collaborative tools, issue tracking, and pull requests. It acts as a social coding platform, enabling sharing and collaboration on projects.

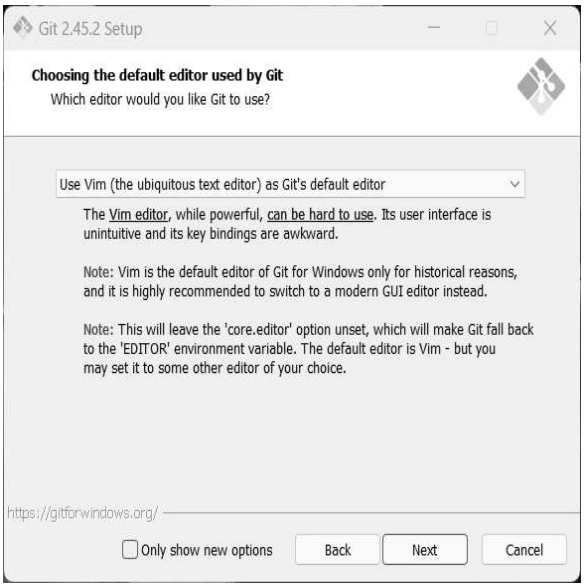
3. **Explain Different Git Commands**
Answer:

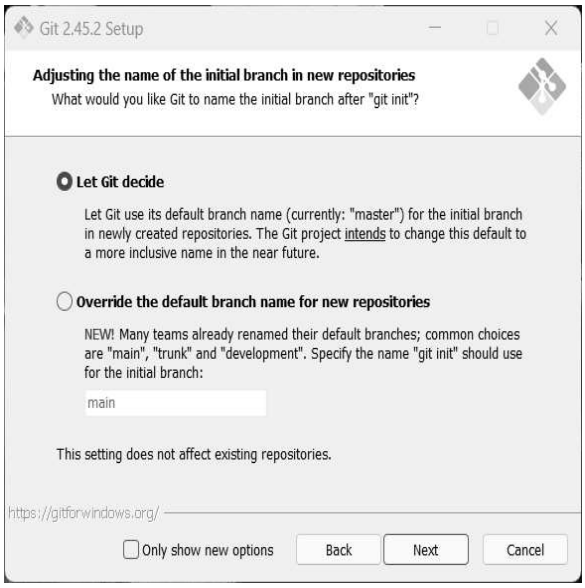
- git init: Initializes a new Git repository.
- git clone [URL]: Copies an existing repository.
- git add [file]: Stages changes for the next commit.
- git commit -m "[message]": Commits staged changes with a message.
- git status: Shows the current state of the working directory.
- git diff: Displays differences between files.
- git branch [name]: Creates a new branch.
- git checkout [name]: Switches to a different branch.
- git merge [branch]: Merges another branch into the current one.
- git fetch [remote]: Downloads changes from a remote repository.
- git pull [remote] [branch]: Fetches and merges changes from a remote branch.
- git push [remote] [branch]: Pushes local commits to a remote repository.
- git reset [commit]: Resets the current branch to a specific commit.
- git revert [commit]: Creates a commit that undoes a previous commit.
- git log: Shows the commit history.
- git show [commit]: Displays detailed information about a specific commit.

B. Execute following on Git and GitHub (Refer the shared material) and attach screenshots:

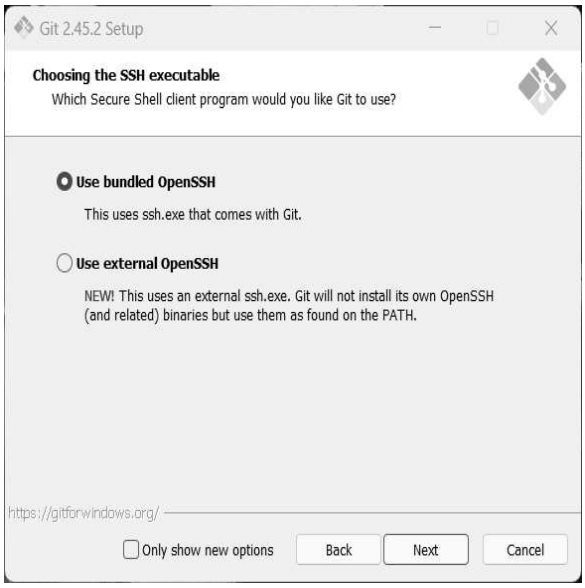
Git Installation:



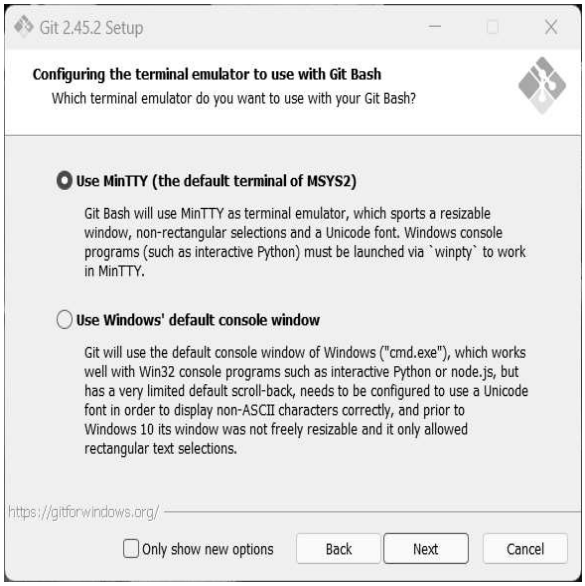

Git 2.45.2 Setup
Choosing the default editor used by Git
Which editor would you like Git to use?
Use Vim (the ubiquitous text editor) as Git's default editor
The Vim editor, while powerful, can be hard to use. Its user interface is unintuitive and its key bindings are awkward.
Note: Vim is the default editor of Git for Windows only for historical reasons, and it is highly recommended to switch to a modern GUI editor instead.
Note: This will leave the 'core.editor' option unset, which will make Git fall back to the 'EDITOR' environment variable. The default editor is Vim - but you may set it to some other editor of your choice.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

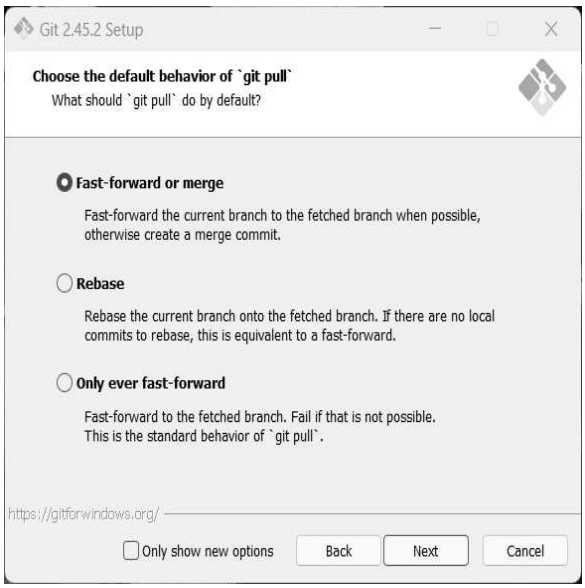

Git 2.45.2 Setup
Adjusting the name of the initial branch in new repositories
What would you like Git to name the initial branch after "git init"?
Let Git decide
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project intends to 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/
Only show new options
Back
Next
Cancel

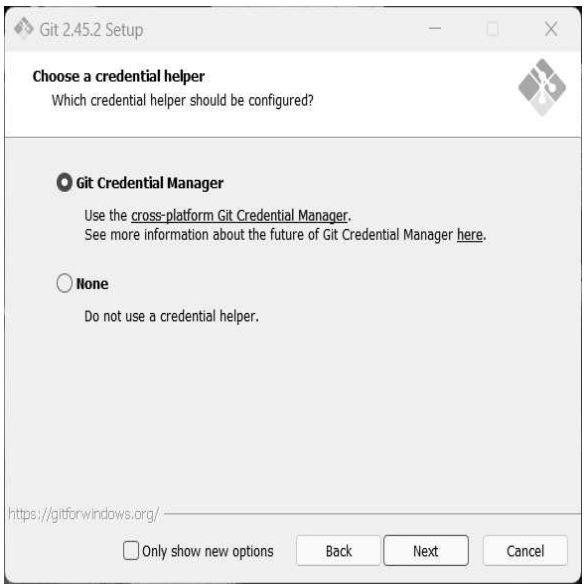

Git 2.45.2 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the most cautious choice as your PATH will not be modified at all. You will only be able to use the Git command line tools from Git Bash.
Git from the command line 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/
Only show new options
Back
Next
Cancel

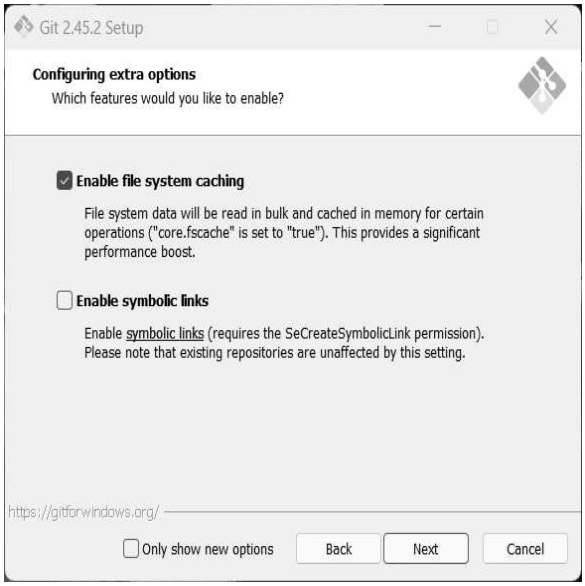

Git 2.45.2 Setup
Choosing the SSH executable
Which Secure Shell client program would you like Git to use?
Use bundled OpenSSH
This uses ssh.exe that comes with Git.
Use external OpenSSH
NEW! This uses an external ssh.exe. Git will not install its own OpenSSH (and related) binaries but use them as found on the PATH.
https://gitforwindows.org/
Only show new options
Back
Next
Cancel

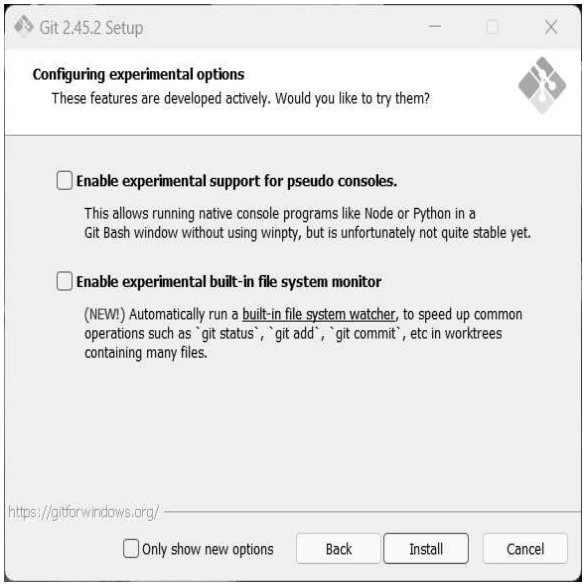

Git 2.45.2 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, 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/
Only show new options
Back
Next
Cancel

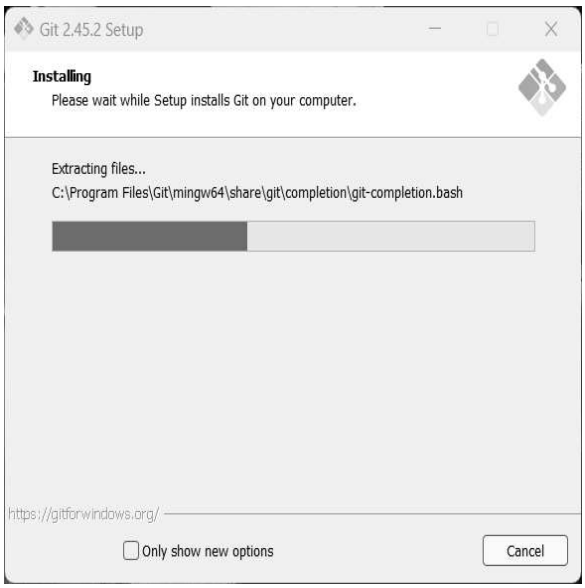

Git 2.45.2 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. 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/
Only show new options
Back
Next
Cancel


Git 2.45.2 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/
Only show new options
Back
Next
Cancel


Git 2.45.2 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/
Only show new options
Back
Next
Cancel


Git 2.45.2 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/
Only show new options
Back
Next
Cancel


Git 2.45.2 Setup
Configuring experimental options
These features are developed actively. Would you like to try them?
Enable experimental support for pseudo consoles.
This allows running native console programs like Node or Python in a Git Bash window without using winpty, but is unfortunately not quite stable yet.
Enable experimental built-in file system monitor
(NEW!) Automatically run a built-in file system watcher, to speed up common operations such as `git status`, `git add`, `git commit`, etc in worktrees containing many files.
https://gitforwindows.org/
Only show new options
Back
Install
Cancel

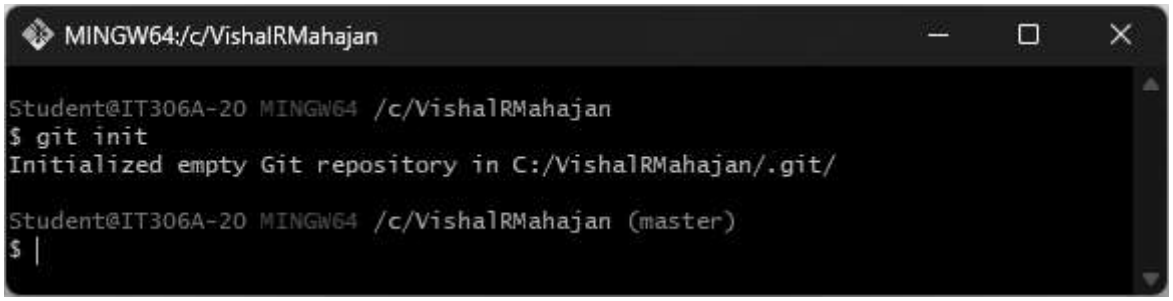

Git 2.45.2 Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
C:\Program Files\Git\mingw64\share\git\completion\git-completion.bash
https://gitforwindows.org/
Only show new options
Cancel

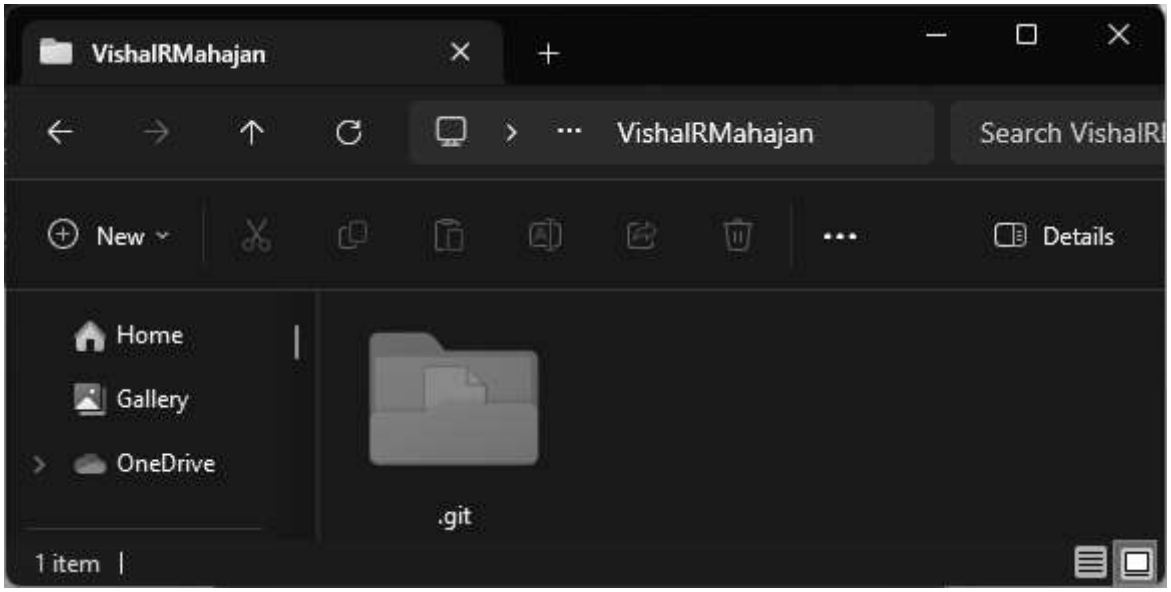

Git 2.45.2 Setup
Completing the Git Setup Wizard
Setup has finished installing Git on your computer. The application may be launched by selecting the installed shortcuts.
Click Finish to exit Setup.
Launch Git Bash
View Release Notes
Only show new options
Finish

Git Commands:

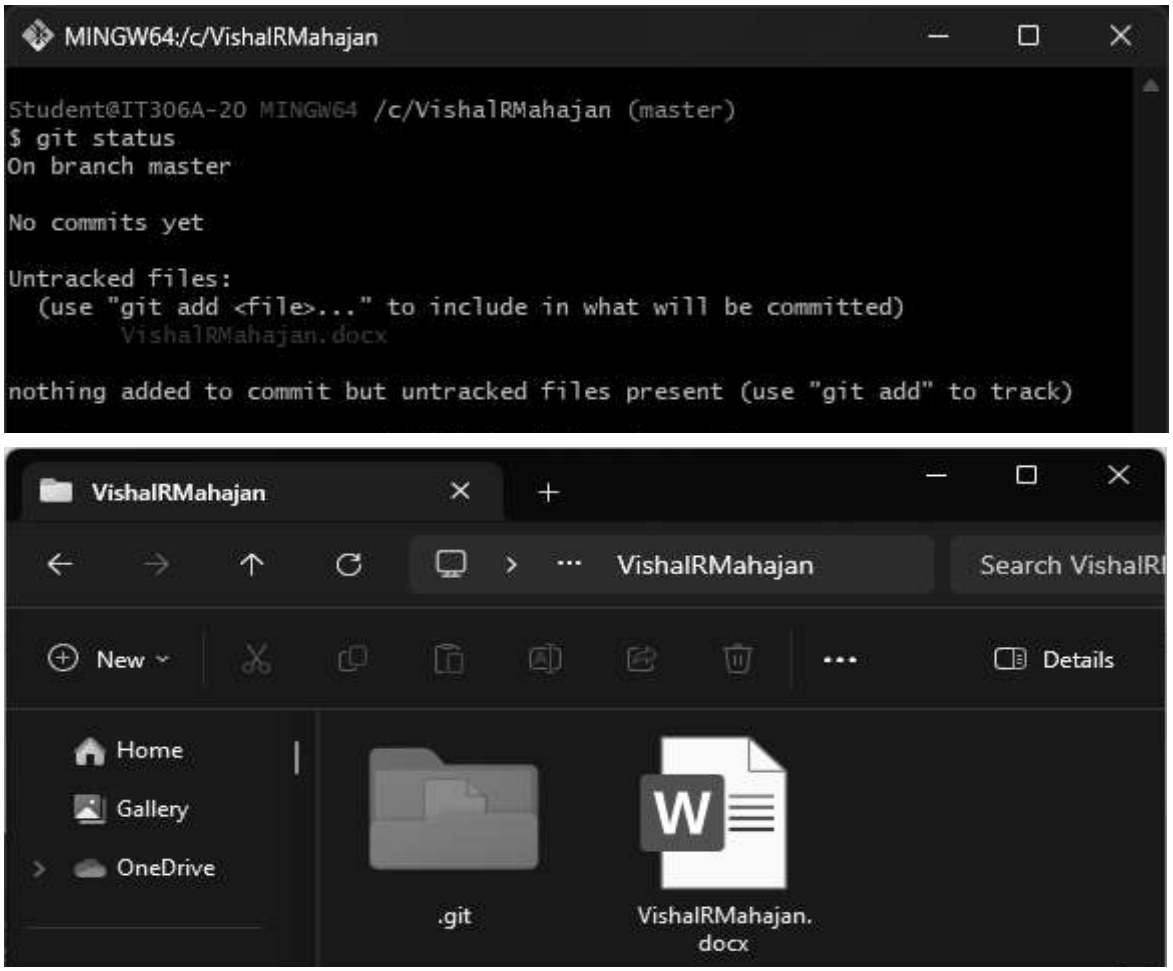
1. **Command**: Git --version
   Screenshot:

2. **Command**: git init
   Screenshot:

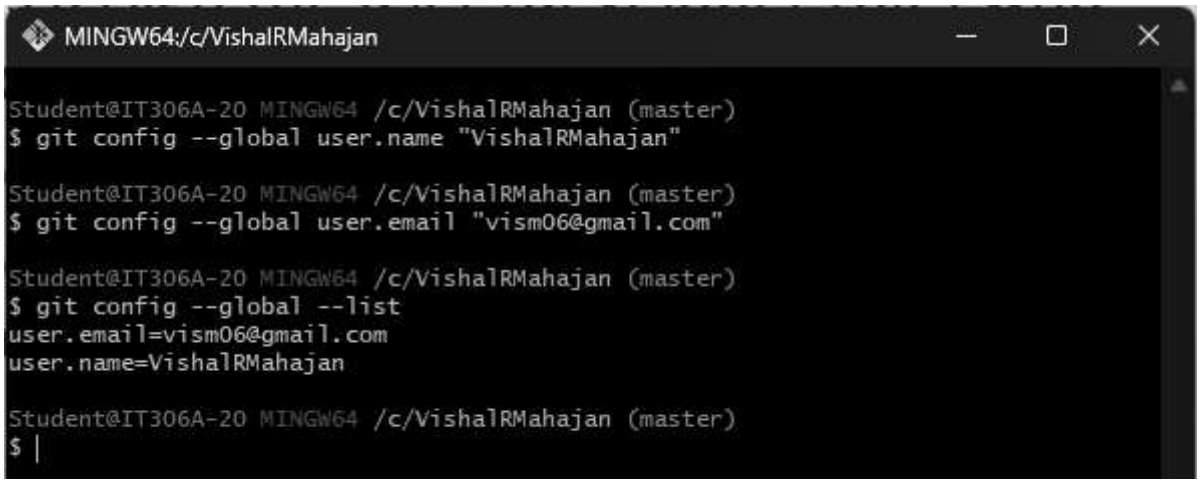




3. **Command**: git status
   Screenshot:



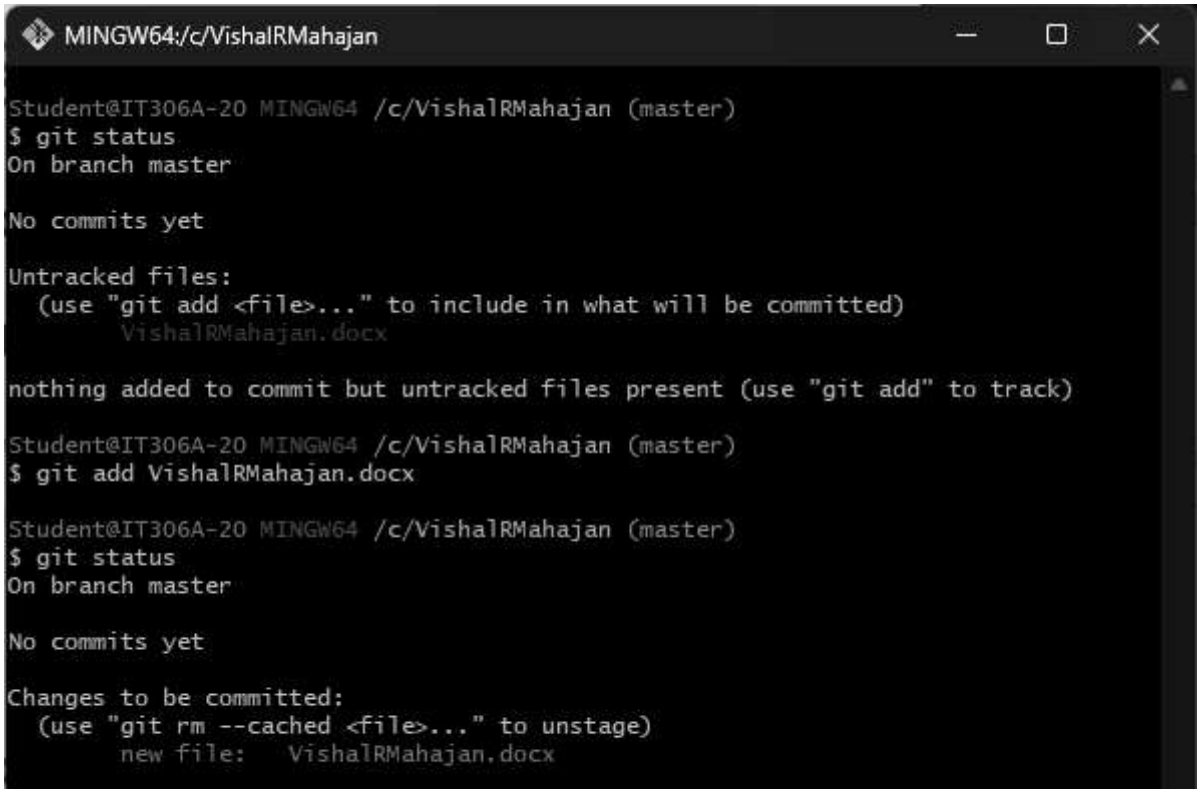
4. **Command**: git config
   Screenshot:

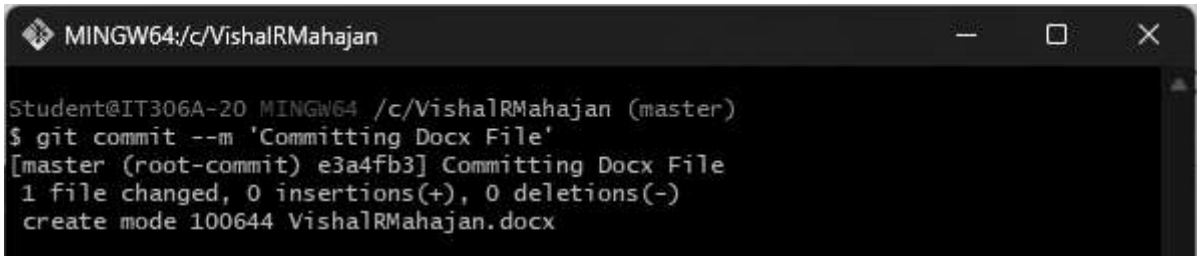


5 & 6. **Command**: git add and git commit

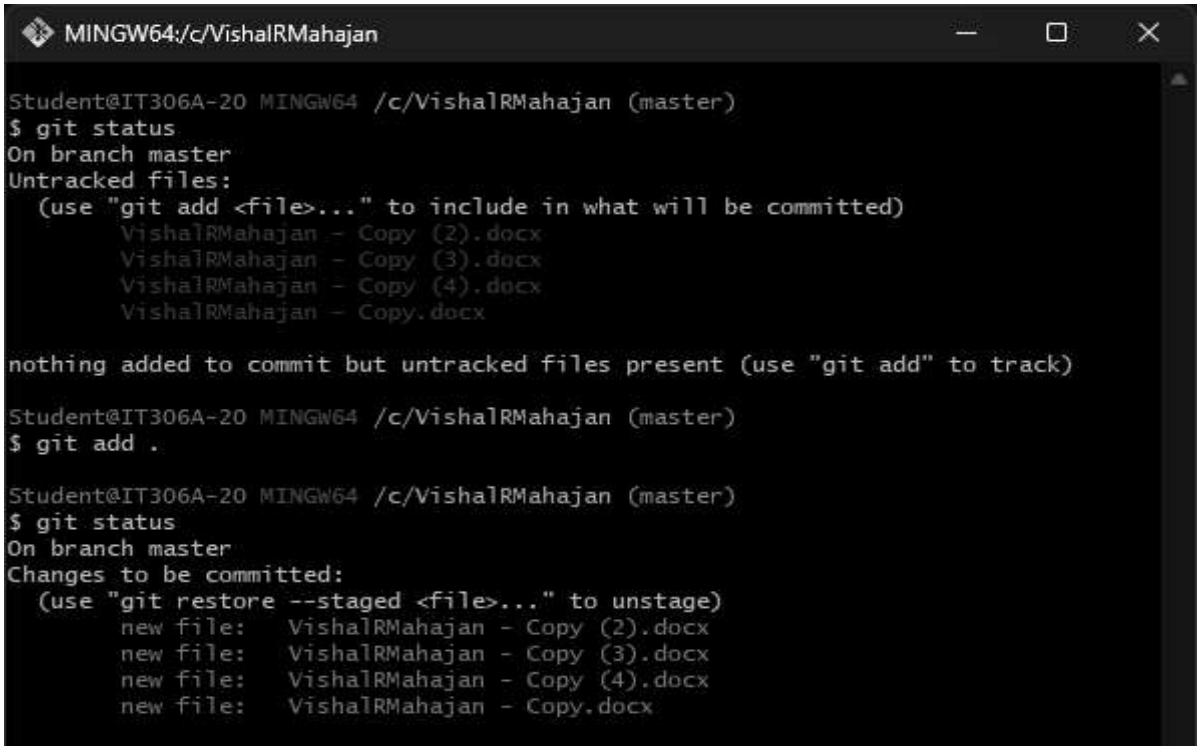
Screenshot:

1. Adding only one file using **git add <filename>** and then committing it using **git commit --m <commit message>**





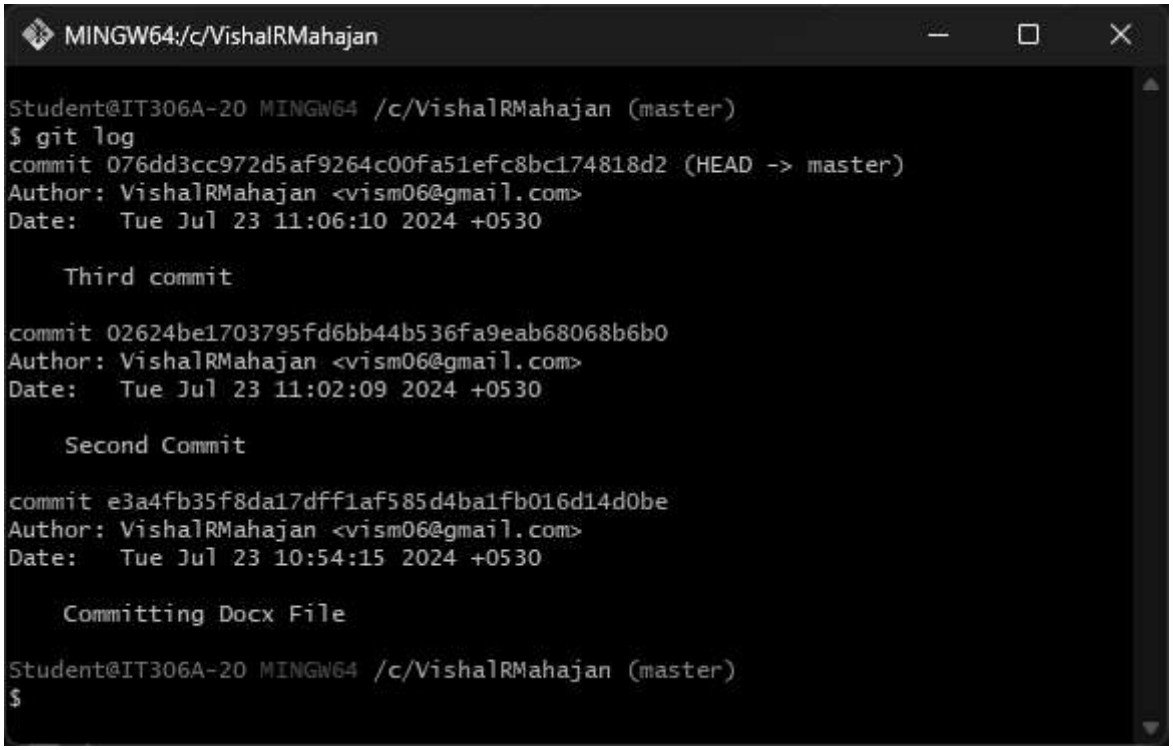
2.Adding all files at once using **git add .** and then committing it using **git commit --m <commit message>**



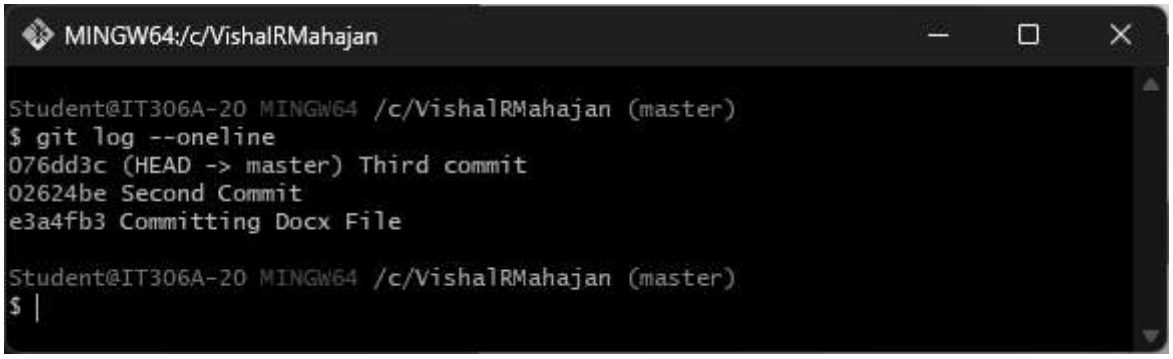
7. Command: git log

Screenshot:

1. Getting the log of all the files committed using **git log**



2. Getting the log of all the files committed in one line using **git log --oneline**



3. Getting log of a particular file in one line using **git log --oneline <filename>**

4. Getting log of a particular file in one line using **git log --oneline <commit hash>**

GitHub Account Creation: