**SOURCE CODE MANAGEMENT**

**COURSE CODE: CSE2015**

**SLOT: L3, L4**



**NAME:** SHREYAS RAICHUR

**ENROLLMENT NUMBER:** A86605224016

**FACULTY:** DR. MONIT KAPOOR

# INDEX

| **S. No.** | **Lab Session Title** | **Page No.** |
| --- | --- | --- |
| **1** | **Git Fundamentals** | **3** |
| **2** | **Installing Git on Windows** | **4** |
| **3** | **Basic CLI Commands** | **5** |
| **4** | **Vim Text Editor** | **8** |
| **5** | **Git Commands** | **10** |
| **6** | **SCM Project (Creating and Managing Repositories)** | **13** |
| **7** | **Team Collaboration (Fork, Clone, Pull Request)** | **18** |

**Lab Session 1: Git Fundamentals**

**Computer**

A **computer** is any device capable of performing calculations, whether they are logical or mathematical.

**Program/Code**

A **program** (or **code**) is a set of instructions, often organized as an algorithm, that directs a computer to perform a specific task.

**Need for Managing Source Code**

Modern applications, such as Spotify, consist of multiple programs working together on both the frontend and backend to deliver smooth user experience. Regular updates are essential for:

* **Fixing Bugs:** Quickly resolving errors that may occur.
* **Improving UI/UX:** Enhancing the user interface and overall experience.
* **Optimizing Performance:** Addressing and refining issues for better performance.

For programmers, effective management of source code is crucial because:

* It ensures that all files remain in context throughout the lifecycle of the program.
* It facilitates collaboration, allowing multiple developers to work together on a shared codebase.

**Tools for Source Code Management**

### Git:

A version control system that runs locally on your computer. Git helps track changes and manage versions of your project.

### GitHub:

A global, cloud-based platform that hosts Git repositories, enabling developers to share, collaborate, and contribute to projects from anywhere in the world.

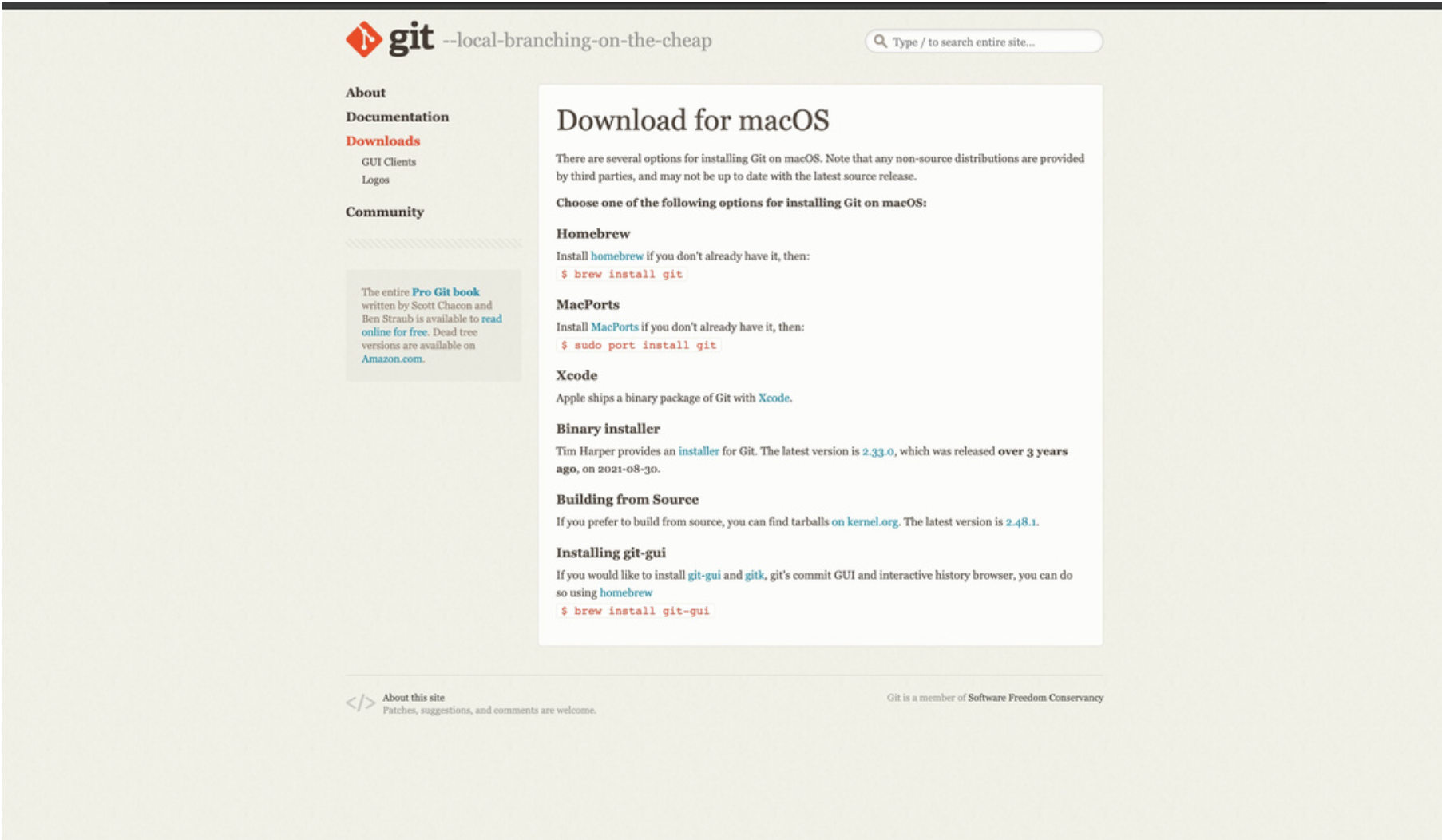
**Version**

A **version** in version control represents a snapshot of your project at a specific moment in time. This snapshot allows you to review, revert, or compare changes made throughout the development process.

# Lab Practical 1

## Installing Git in Windows

Step 1: Visit section 1.5 of pro git document and navigate to macOS section:



Step 2: Install Homebrew (if not already installed):

• Command: /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

•Description: Installs Homebrew, a package manager for macOS.



Step 3: Install Git Using Homebrew:

• Command: brew install git

•Description: installs Git via Homebrew

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Step 4; Verify Git Installation:

# Basic CLI Commands

## 1) Command: pwd

**Description:** Prints the directory the user is working in.

## 2) Command: ls

**Description:** Lists all files and directories in the current directory.

## Command: date

Description: shows the current date and time in a standard format

## Command: clear

Description: The clear command in the CLI is used to clear all the current text and output displayed in the terminal window.

## Command: time

Description: The time command in the CLI is used to measure the execution time of a command or program.

## Command: cd ‘Directory’

**Description:** Changes the current working directory to the desired directory.

## Command: cd ..

Description: Goes back to the previous directory.

## Command: mkdir

**Description:** To create a new directory.

## 9. Command: rmdir

**Description: To delete a directory**



# 3. Vim Text Editor

## Command: vi hello.txt

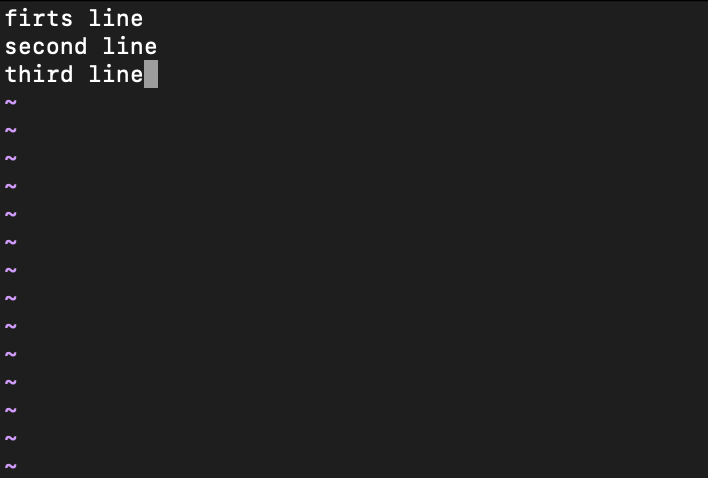
**Description:** Opens (or creates) the file hello.txt in the Vim text editor.



## Command: i (Insert Mode)

**Description:** Enters insert mode in Vim to allow text input.





## Command: esc

Description: Used to exit insert mode

A black screen with white text

AI-generated content may be incorrect.

1. **Command: :wq**

**Description:** Saves the changes and exits the Vim editor.

A black screen with a white text

AI-generated content may be incorrect.

# 4. Git Commands

## Command: git - - version

Description: The git --version command is used to check the installed version of Git on your system.

bash

1. **Command: git init**

**Description:** Initializes a new Git repository in the current directory.

1. **Command: git status**

**Description:** Displays the current status of the working directory and staging area.

**4. Command: git add Test.c**

**Description:** Add Test.c to the staging area in preparation for a commit.

**5. Command: git commit -m “add file one”**

**Description:** Commits the stage changes with the message “add file one”.

**6. Command: git log**

**Description:** Display the commit history of the repository.

**7. Command: git clone**

**Description:** To obtain a copy of an existing Git repository.

**8. Command: git log --oneline**

**Description:** For generating shorter commit ID.

**9. Command: git diff**

**Description:** To compare two files.

**10. Command: git remote add “Variable”**

**Description:** To connect with the Users GitHub account.

**11. Command: git remote**

**Description:** To check the status of the repositories connected with the Users account.

**12. Command: git push -u “Variable” master**

**Description:** To push all the files to the Users account.

**13. Command: git merge “File\_Name” -m “comment”**

**Description:** To merge a branch with main branch.

**SCM Project**

The project was to make a repository in GitHub, make 3 branches and merge it with the main branch and access all 4 team-mate’s repositories, fork it, clone it, make some changes and merge them.

First, make your own repositories and make 3 branches and add files and merge with the main branch.

1. Go to the directory on your computer

1. Clone the GitHub repositories

1. Change to the folder

1. Make an initial commit as a README.md and commit it

1. Make four branches

1. Checkout to the first branch, add the files and make two commits

Checkout to the second branch, add the files and make two commits

1. Checkout to the third branch, add the files and make two commits

1. Checkout to the main branch

1. Merge all the branches to the main branch
2. Push all the commits and files in the Github repository

*\*(Now we will clone the repository of the rest of my Team Members, make some changes in their repositories, and send pull requests.)*

* Dominic

1)Go back to the folder where you want to save the folder

2)Clone the forked repository

3)Go to the cloned repository

4)Create a new branch and checkout to it

5)Make some changes in the repository.

6)Add the edited files.

7)Commit the files.

8)push it to the Forked Repository for pull request.

* Sanin

1)Go back to the folder where you want to save the folder

2)Clone the Forked repository

3)Go to the cloned repository.

4)Create a new branch and checkout to it.

5)Make changes in the file

6)Add the files and commit the files and Push the branch to the forked repository for pull request.

* Aneesh

1)Go back to the folder where you want to save the folder

2)Clone the Forked Repository.

3)Go to the cloned repository.

4)Create a branch and checkout to it.

5) Make changes to the files.

6)Add the files and then commit it and Push the branch to the forked repository for pull request.

*\*(Finally, after all the changes made and pull requests we use “git pull” to sync all the changes in our local folder)*

8) Finally after all work in github.  
  
we update the pull request.

9) git merge dev

10) git push origin main

1. Git push origin de