NAME: MEGHA.M

SEN: A86605224150

SOURCE CODE MANAGEMENT:

LAB MANUAL

❖ Introduction- GitBash

Git Bash is a command-line interface for Windows that provides an emulation layer for a Unix-like terminal experience. It is part of the Git for Windows package and allows users to interact with Git repositories using Bash commands.

Key Features of Git Bash

• Unix-like Environment: Git Bash provides a Bash shell, which is commonly used in Linux and macOS.

• Git Integration: It includes Git commands, allowing users to manage repositories efficiently.

• Command-Line Utilities: Comes with essential Unix commands like ls, cd, pwd, grep, and more.

• File Navigation: Users can navigate directories and execute Git operations seamlessly.

How to Install Git Bash

1. Download Git for Windows from Git's official website.

2. Run the installer and follow the setup instructions.

3. Choose Git Bash as the default terminal option.

4. Once installed, open Git Bash and start using Git commands.

Basic Git Bash Commands

- **git --version → Check installed Git version.**
- **git init → Initialize a new Git repository.**
- **git clone → Clone an existing repository.**
- **git status → Check the status of changes.**
- **git add → Stage changes for commit.**
- **git commit -m "message" → Commit changes with a message.**
- **git push → Push changes to a remote repository.**

Git Bash is a great tool for developers who prefer a Unix-like terminal on Windows while working with Git repositories. You can explore more details in this tutorial. Let me know if you need help with anything specific!

❖ GitBash and Github

❖ Git Bash and GitHub are both essential tools for version control and collaboration in software development, but they serve different purposes.

Git Bash:

Git Bash is a command-line interface for Windows that provides a Unix-like environment for using Git. It allows developers to execute Git commands and interact with repositories using Bash shell commands.

GitHub:

GitHub is a cloud-based platform that hosts Git repositories and provides collaboration tools for developers. It allows teams to work on projects together, track changes, and manage code efficiently.

How They Work Together

1. Local Development: Developers use Git Bash to manage repositories on their local machines.

2. Version Control: Git Bash helps track changes using Git commands (git add, git commit, git push).

3. Remote Collaboration: GitHub stores repositories online, enabling multiple developers to contribute.

4. Pull Requests & Issues: GitHub provides features like pull requests and issue tracking for project management.

If you're looking for a detailed comparison, you can check out this discussion or download Git from here

❖ File Creation with commit and push command

❖ To create a file, commit it, and push it to a remote repository using Git Bash, follow these steps:

1. Create a File

Open Git Bash and navigate to your repository folder:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ cd /path /to/your / repository
```

Create a new file using:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ vi filename.extension
```

The vi command in Linux is used to open and edit files using the Vi editor, a powerful text editor available on most Unix-based systems.

Basic Vi Commands

- **Insert Mode:**
  Press i to start editing.
- **Save and Exit:**

Press Esc then type :wq to save and exit.
Use :q! to exit without savin

2. Add the File to Git:

- Stage the file for commit:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git add filename.txt
```

To add all files in the directory:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git add .
```

3. Commit the File

Commit the changes with a message:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git commit -m "added filename.txt"
```

4. Push to Remote Repository

Push the changes to GitHub (assuming origin is the remote and main is the branch):

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git push origin master
```

Shortcut: Add, Commit, and Push in One Command

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git push origin master
```

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git push origin master
```

Branches Creation

In Git, branches allow developers to work on different features or fixes without affecting the main codebase. Here's how you can create and manage branches:

o Creating a New Branch

To create a new branch, use:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git branch branch_name
```

o Switching to a Branch

To switch to the newly created branch:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git checkout  branch_name
```

Listing All Branches

To see all branches in your repository:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git branch
```

Deleting a Branch

After merging, you can delete the branch:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git branch -d branch_name
```

❖ Merge Request

❖ A Merge Request (MR) is a feature used in Git-based platforms like GitLab to propose and review changes before merging them into the main branch. It is similar to a Pull Request (PR) in GitHub.

How a Merge Request Works

1**. Create a Branch** – Developers create a new branch for their changes.
2. **Make Changes** – Code modifications are made in the branch.
3. **Open a Merge** Request – The developer submits a request to merge the branch into the main branch.

4. **Code Review** – Team members review the changes, suggest improvements, and approve the request**.**

5. Merge the Branch – Once approved, the branch is merged into the main codebase.

Steps to Merge a Branch

1. **Switch to the Target Branch (usually main or master):**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git checkout main
```

2. **Merge the Branch:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git merge branch_name
```

Key Benefits

- **Ensures code quality through peer reviews.**
- **Helps track changes before merging.**
- **Allows collaboration among developers.**

❖ Open and Close Pull Request

❖ A Pull Request (PR) is a way to propose changes to a repository before merging them. Here's how you can open and close a pull request on GitHub:

Opening a Pull Request

1. **Push your changes to a new branch:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git push origin  branch_name
```

2. Go to GitHub and navigate to your repository.
3. Click on the Pull Requests tab and select New Pull Request.
4. Choose the branch you want to merge into the main branch.
5. Add a title and description, then click Create Pull Request.

Closing a Pull Request

- **If you want to close a pull request without merging:**

1. Open the pull request on GitHub.
2. Scroll down and click Close Pull Request.
3. Optionally, delete the branch to keep your repository clean.

❖ Complete Git Process

1. Install Git Bash

- Download Git for Windows from Git's official website.
- Run the installer and follow the setup instructions.
- Choose Git Bash as the default terminal option.

2. Initialize a Repository

- Open Git Bash and navigate to your project folder:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ cd /path /to/your / repository
```

Initialize a new Git repository:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git init
```

3. Create and Modify Files

- Create a new file:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ vi filename.extension
```

4. Check Repository Status

- View changes:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git status
```

5. Stage and Commit Changes

- **Add files to the staging area**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git add filename.txt
```

- **Commit changes:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git commit -m "added filename.txt"
```

6. Create and Manage Branches

- **Create a new branch:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git branch branch_name
```

- **Switch to the branch:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git checkout  branch_name
```

- **Merge the branch into the main branch:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git merge branch_name
```

7. Push Changes to GitHub

- **Add a remote repository:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git remote add origin <repo-url>
```

- **Push changes:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git push origin master
```

8. Pull Changes from Remote Repository

- **Update local repository:**

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git push origin master
```

9. Open and Close Pull Requests

- Push your branch to GitHub:
- Open a Pull Request on GitHub and merge changes.
- Close the Pull Request if needed.

10. Resolve Merge Conflicts

If conflicts occur, manually edit the files, then:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git add complicated_file
```

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git commit -m "resolved merge conflict"
```

11. Delete a Branch

- After merging, delete the branch:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git branch -d branch_name
```

Note: Git clone

The git clone command is used to create a local copy of a remote Git repository. It downloads all files, branches, and commit history, allowing you to work on the project locally.

Basic Usage

To clone a repository, use:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git clone <repository-url>
```

This creates a local copy of the repository in a new directory.

Cloning a Specific Branch

If you want to clone only a specific branch:

```
Megha@LAPTOP-6NMIT37V MINGW32 ~ (master)
$ git clone -b branch_name --single-branch <repository-url>
```

❖ BASICS OF LINUX :

1. What is Linux?

Linux is an open-source operating system based on the Linux kernel, first developed by Linus Torvalds in 1991. It is known for its stability, security, and flexibility.

2. Linux Distributions (Distros)

Linux comes in various distributions (distros), which are different versions tailored for specific needs.

Some popular ones include:

- **Ubuntu (User-friendly, great for beginners)**
- **Fedora (Cutting-edge features)**
- **Debian (Stable and reliable)**
- **Arch Linux (Highly customizable)**
- **CentOS (Enterprise-focused)**

3. The Linux Terminal

Unlike Windows, Linux relies heavily on the command line interface (CLI). Some essential commands include:

- **ls – Lists files in a directory**
- **cd – Changes directories**
- **mkdir – Creates a new directory**
- **rmdir – Deletes files or directories**
- **sudo – Runs commands with administrative privileges**

4. File System Structure

Linux has a hierarchical file system:

- **/home – User files**
- **/bin – Essential binaries**
- **/etc – Configuration files**
- **/var – Variable data (logs, caches)**
- **/tmp – Temporary files**

5. Package Management

Linux uses package managers to install software:

- **APT (apt-get install) for Debian-based distros**
- **YUM/DNF (yum install or dnf install) for Red Hat-based distros**
- **Pacman (pacman -S) for Arch Linux**