****

**SOURCE CODE MANAGEMENT**

## **LAB FILE**

***Name – Meghana. S. A.***

***SEN – A866185724005***

***Slot – L5+L6***

***Submitted to – Monit Kapoor Sir***

***(Pro VC Amity University)***

Contents

[**LAB FILE 1**](#_Toc200302743)

[**Program 1: Installation of Git Bash 4**](#_Toc200302744)

[**Program 2: Configure GitHub in Git Bash 5**](#_Toc200302745)

[**Program 3: Initialize a Git Repository 6**](#_Toc200302746)

[**Program 4: Check Current Branch (master) 7**](#_Toc200302747)

[**Program 5: Add Files to Git Staging Area 8**](#_Toc200302748)

[**Program 6: Commit Files in Master Branch 8**](#_Toc200302749)

[**Program 7: Create and Switch to dev Branch 9**](#_Toc200302750)

[**Program 8: Add and Commit in dev Branch 9**](#_Toc200302751)

[**Program 9: Connect Local Repo to Remote GitHub Repo 10**](#_Toc200302752)

[**Program 10: Push dev and master Branch to GitHub 10**](#_Toc200302753)

[**Program 11: Merge dev into master 12**](#_Toc200302754)

[**Program 12: Fork a Repository on GitHub 12**](#_Toc200302755)

[**Program 13: Clone Forked Repository 13**](#_Toc200302756)

[**Program 14: Create Feature Branch 14**](#_Toc200302757)

[**Program 15: Commit in Feature Branch 14**](#_Toc200302758)

[**Program 16: Push Feature Branch 15**](#_Toc200302759)

[**Program 17: Open Pull Request 15**](#_Toc200302760)

[**Program 18: Merge Pull Request 16**](#_Toc200302761)

[**Program 19: Pull Latest Changes from GitHub 16**](#_Toc200302762)

# Program 1: Installation of Git Bash

**Aim:**  
To install Git Bash on a Windows system for version control operations.

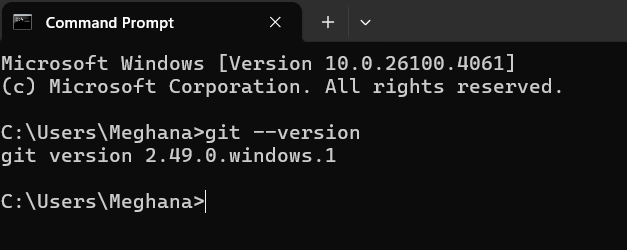
**Theory:**  
Git Bash is an application that provides Git command line features along with a Bash emulator. It helps in executing Git commands and using Unix-style commands on Windows.

**Software Requirements:**

* Windows OS
* Internet Connection

**Procedure:**

1. Visit the official Git website: <https://git-scm.com/>
2. Click on **"Download for Windows"**.
3. Run the downloaded .exe file.
4. Follow the installation wizard:
   * Choose default options unless specific preferences are needed.
   * Select “Use Git from the command line and also from 3rd-party software”.
5. Finish installation and launch **Git Bash**.
6. Check the version using the command:



**Result:**  
Git Bash was successfully installed and launched.

# Program 2: Configure GitHub in Git Bash

**Aim:**  
To configure GitHub account details in Git Bash for using Git version control.

**📚 Theory:**

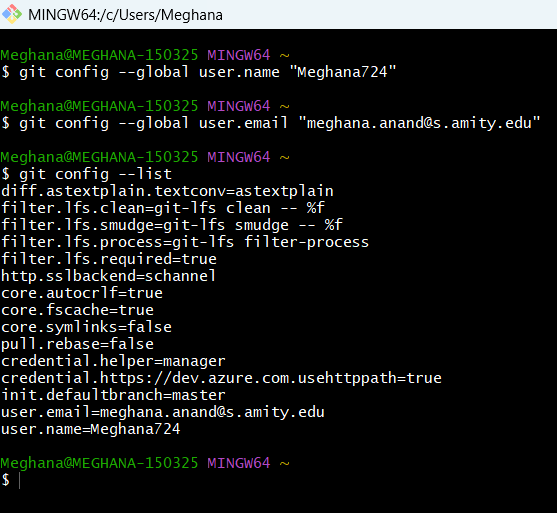
Git needs to know who you are so that it can track who made which changes. To use GitHub with Git Bash, we need to set the global username and email, which should match your GitHub account.

**🛠️ Software/Tool Requirements:**

* Git Bash (already installed)
* GitHub account (created)
* Internet Connection

**📋 Procedure:**

1. **Open Git Bash.**
2. Set your GitHub username by entering the following command:



**Result:**

GitHub account has been successfully configured in Git Bash with username and email.

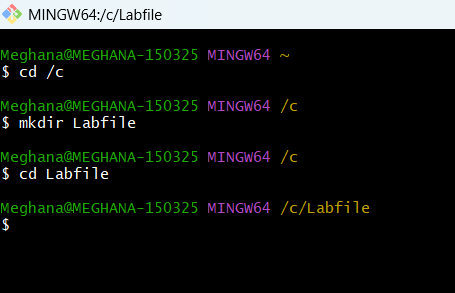
# Program 3: Initialize a Git Repository

**Aim:**  
To initialize a Git repository in a local project directory using Git Bash.

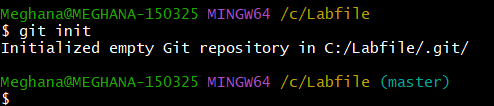
**Theory:**  
Git needs to be initialized in a project folder to start tracking changes using version control. The git init command sets up the folder as a Git repository.

**Procedure:**

1. Open Git Bash.
2. Navigate to your project folder:



1. Initialize Git



**Result:**  
A new Git repository was initialized successfully.

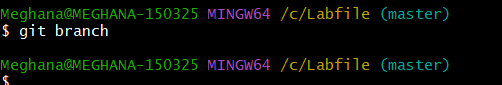
# Program 4: Check Current Branch (master)

**Aim:**  
To verify the active Git branch after initialization.

**Theory:**  
Every Git repository starts with a default branch. In most systems, this is master unless changed manually.

**Procedure:**

1. In the initialized Git repository, run:



**Result:**  
The current branch is displayed as master.

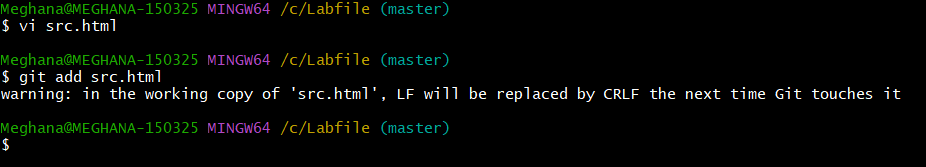
# Program 5: Add Files to Git Staging Area

**Aim:**  
To stage files for the next commit in Git.

**Theory:**  
Before committing, files must be added to the **staging area** using git add. This tells Git which files to track for the upcoming commit.

**Procedure:**

1. Create a file (optional step, like index.html).
2. Run:



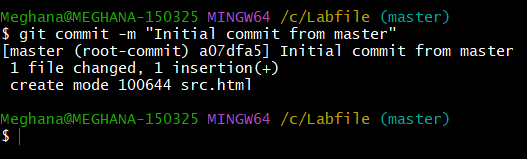
**Result:**  
Files added to the staging area.

# Program 6: Commit Files in Master Branch

**Aim:**  
To create a commit with a meaningful message on the master branch.

**Theory:**  
A commit is a saved version of your code at a specific time. Commits help track changes and roll back if necessary.

**Procedure:**



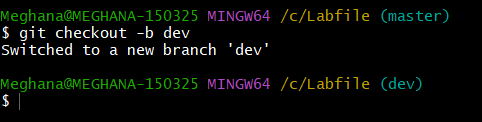
**Result:**  
Files committed successfully to the master branch.

# Program 7: Create and Switch to dev Branch

**Aim:**  
To create and move to a new branch named dev.

**Theory:**  
Branches allow developers to work on features without affecting the main/master branch. This helps in parallel development and testing.

**Procedure:**



**Result:**  
New dev branch created and activated.

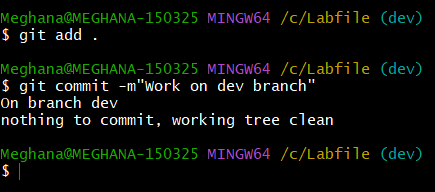
# Program 8: Add and Commit in dev Branch

**Aim:**  
To add and commit changes on the dev branch.

**Theory:**  
Each branch can have its own independent commits. This allows working on updates without touching the master.

**Procedure:**

1. Make or modify a file.
2. Add and commit:



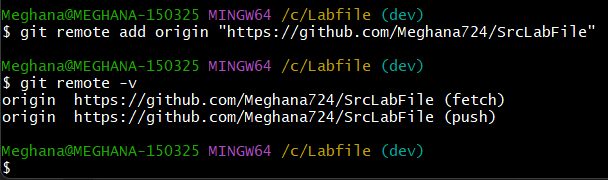
**Result:**  
Changes added and committed in dev branch.

# Program 9: Connect Local Repo to Remote GitHub Repo

**Aim:**  
To add a remote repository link to the local Git project.

**Theory:**  
Remote repositories on GitHub allow code sharing and collaboration. The origin is the name given to the main remote URL.

**Procedure:**



**Result:**  
Remote origin added pointing to GitHub repo.

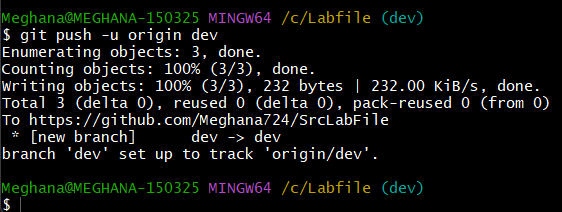
# Program 10: Push dev and master Branch to GitHub

**Aim:**  
To push both dev and master branches from the local repository to the remote GitHub repository.

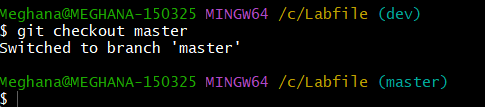
**Theory:**  
Pushing sends your committed work from your local machine to GitHub. It helps back up your work and collaborate with others. You can push multiple branches individually using the git push command.

**Procedure:**

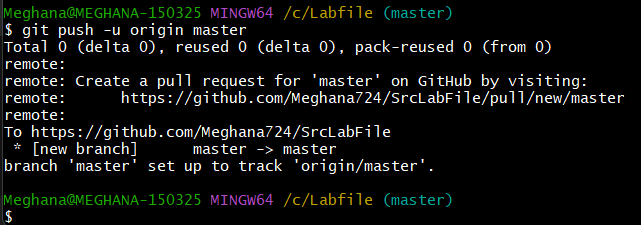
1. ✅ **Push the dev branch**



1. 🔁 **Switch to the master branch**:



1. ✅ **Push the master branch**:



**Result:**  
Both the dev and master branches have been successfully pushed to the GitHub repository.

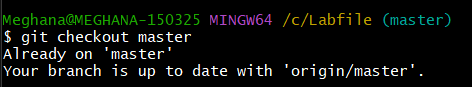
# Program 11: Merge dev into master

**Aim:**  
To merge changes from dev into master.

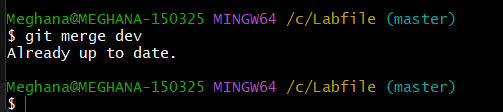
**Theory:**  
Merging brings changes from one branch into another.

**Procedure:**

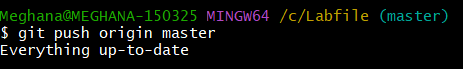
1. Ensure you're on master



1. Merge dev:



1. Push updated master:



**Result:**  
dev merged into master and pushed.

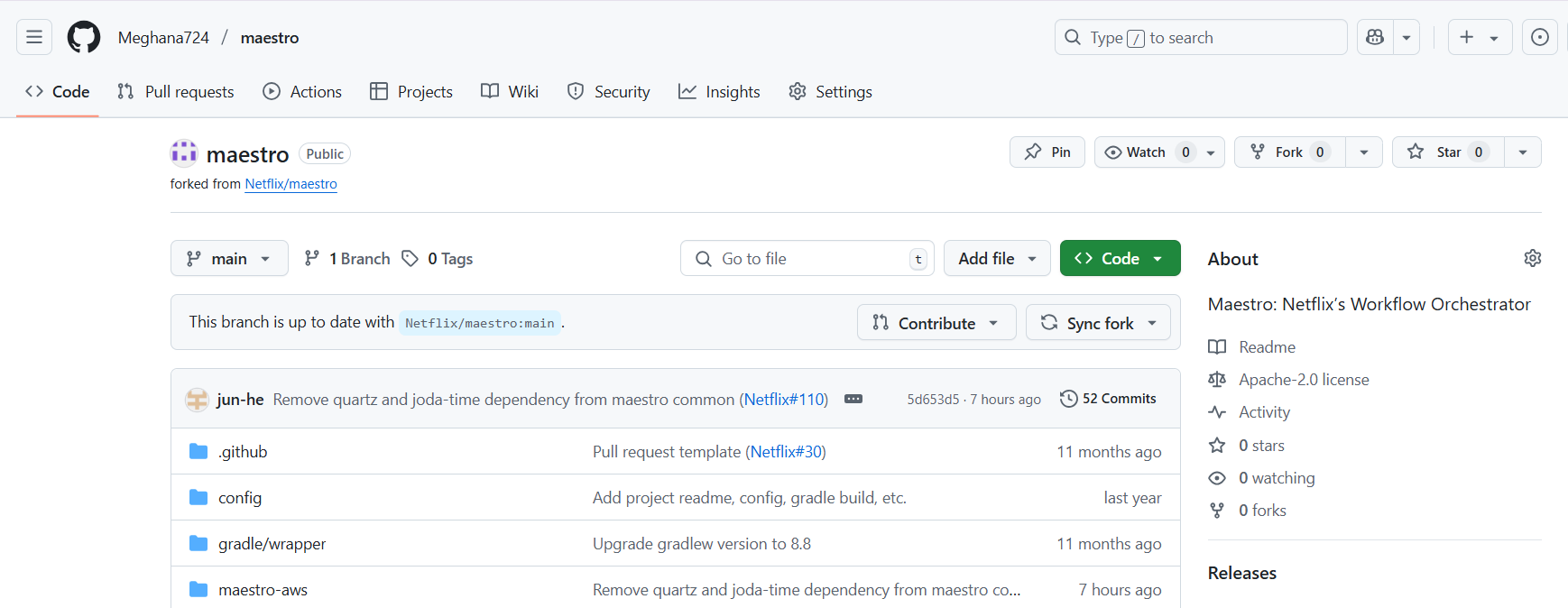
# Program 12: Fork a Repository on GitHub

**Aim:**  
To fork a public GitHub repository.

**Theory:**  
Forking copies a repo into your GitHub account.

**Procedure:**

1. Open the original repo on GitHub.
2. Click **Fork** on the top right.



**Result:**  
Repository forked successfully.

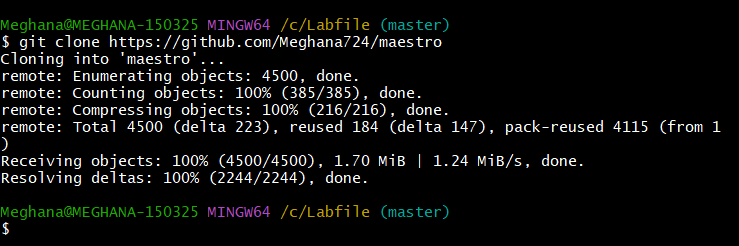
# Program 13: Clone Forked Repository

**Aim:**  
To clone the fork to your system.

**Theory:**  
Cloning brings all project files locally.

**Procedure:**

1. Copy repo URL.
2. Run:



**Result:**  
Repo cloned to local system.

# Program 14: Create Feature Branch

**Aim:**  
To create a new branch for changes.

**Theory:**  
Useful for feature development.

**Procedure:**

1. Run:



**Result:**  
Feature branch created.

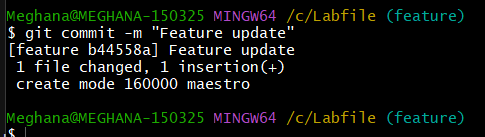
# Program 15: Commit in Feature Branch

**Aim:**  
To commit updates in feature branch.

**Theory:**  
Keeps features separate until ready.

**Procedure:**

1. Make changes.
2. Run:



**Result:**  
Changes committed in feature branch.

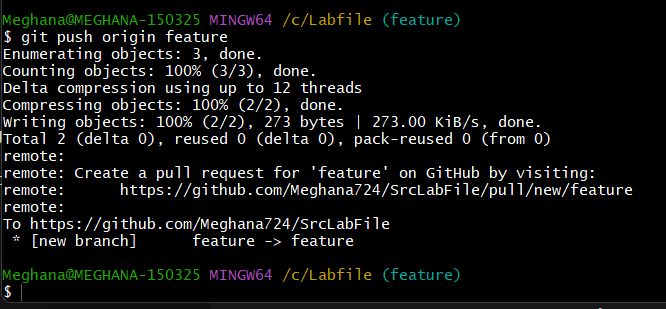
# Program 16: Push Feature Branch

**Aim:**  
To push feature branch to GitHub.

**Theory:**  
Allows others to view/test your changes.

**Procedure:**

1. Run:



**Result:**  
Feature branch pushed.

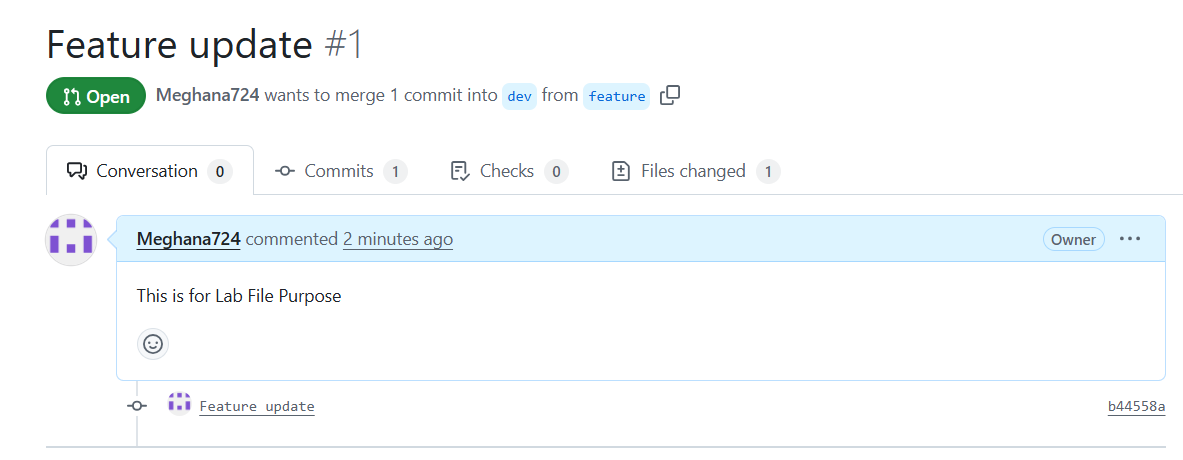
# Program 17: Open Pull Request

**Aim:**  
To request merging feature branch.

**Theory:**  
Pull requests are collaboration tools.

**Procedure:**

1. Go to your repo on GitHub.
2. Click **Compare & Pull Request**.
3. Add title and comment, then click **Create Pull Request**.



**Result:**  
Pull request created.

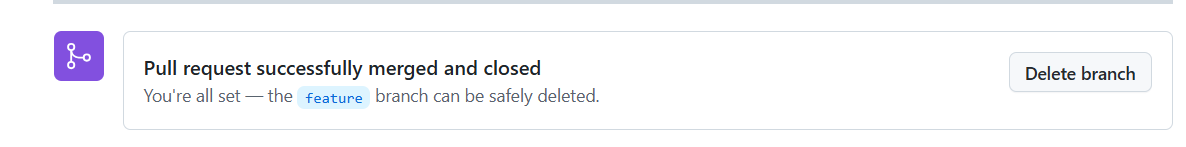
# Program 18: Merge Pull Request

**Aim:**  
To merge a pull request.

**Theory:**  
Allows integrating approved changes.

**Procedure:**

1. Go to the PR.
2. Click **Merge Pull Request** > **Confirm Merge**.



**Result:**  
Pull request successfully merged.

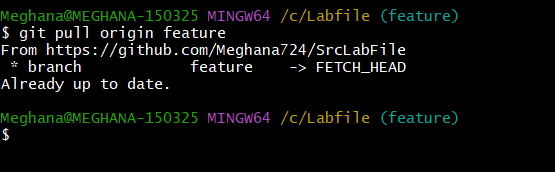
# Program 19: Pull Latest Changes from GitHub

**Aim:**  
To sync local repo with remote.

**Theory:**  
git pull fetches and merges latest changes.

**Procedure:**

1. Run:



**Result:**  
Local repo updated with latest changes.