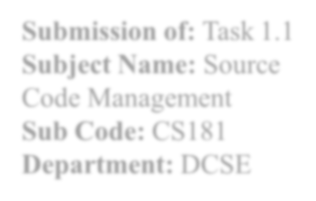
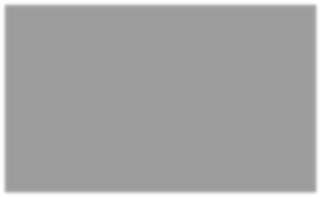
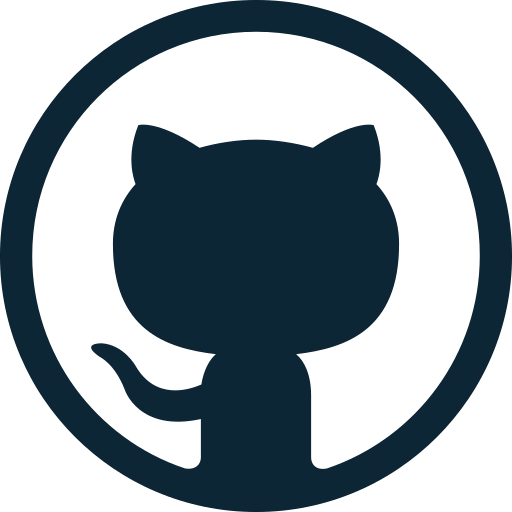
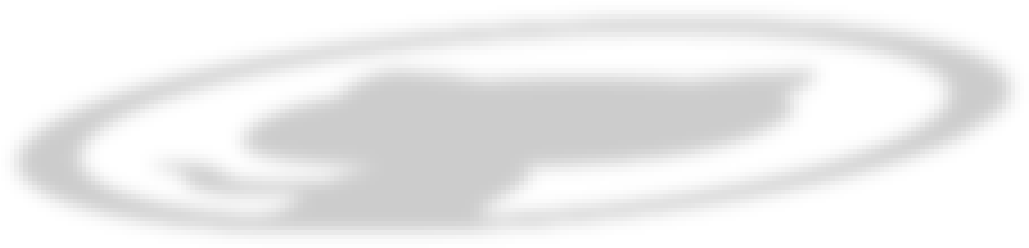
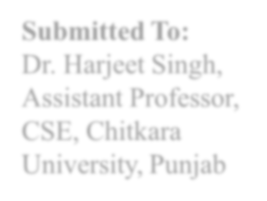
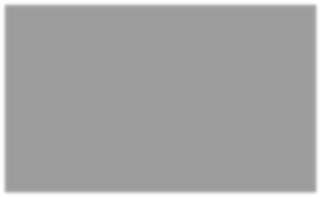
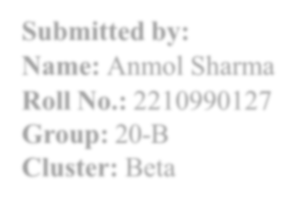
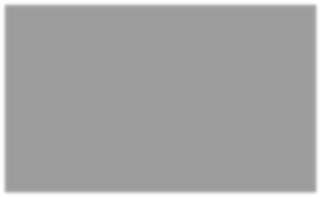


---

**SOURCE CODE**

**MANAGEMENT**

**FILE**



**Submission of:**

Task1.1

**Subject**

**Name:**

Source

Code

Management

**Sub Code:**

003

22

CS

**Department:**

B.E.CSE

**Submitted To:**

Dr. Sharad

Professor,

CSE,

Chitkara

University,

Punjab

**Submitted by:**

**Name:**

Madhav aneja

**Roll**

**No.:**

2310991971

**Group:**

22

-

B

**LIST OF PROGRAMS**

|  |  |  |
| --- | --- | --- |
| **S.**  **No.** | **Program Title** | **Page No.** |
| 1. | Setting Up of Git Client | 3-9 |
| 2. | Setting Up GitHub Account | 10-12 |
| 3. | Generating Logs | 13-15 |
| 4. | Create and Visualize Branches | 16-17 |
| 5. | Git Lifecycle Description | 18-19 |
| 6. |  |  |
| 7. |  |  |
| 8. |  |  |



# Experiment 1

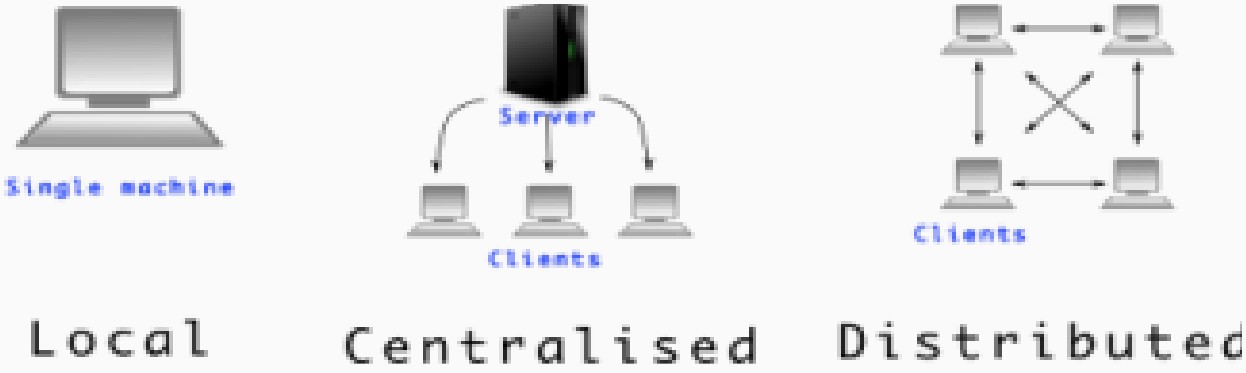
**Aim: Setting up of Git Client**

**What is Git?**

GIT is a Version Control System (VCS). It is a software or we can say a server by which we are able to track all the previous changes in the code. It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can make, edit, recreate, copy or download any code on git hub using git.

**History of VCS:**

* **Local VCS-** No internet Is needed because it uses a database to keep track of files.
* **Centralized VCS-** Centralized version control systems are based on the idea that there is single “central” copy of you project somewhere (probably on the server), and programmers will “commit” their changes to this central copy. “Committing” a change simply means recording the change in the central system.
* **Distributed VCS-** A type of version control where the complete codebase including its full version history is mirrored on every developers computer.

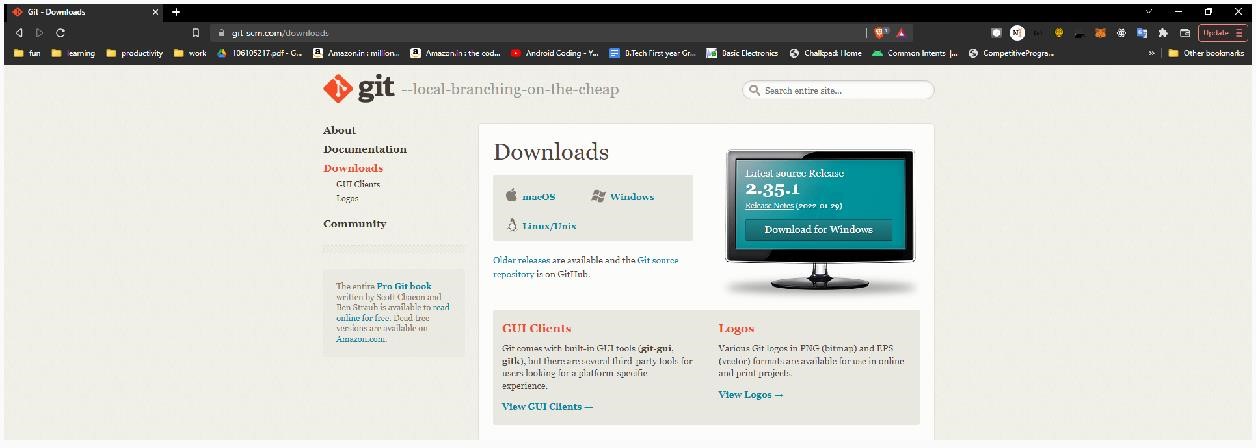


*Figure 1*

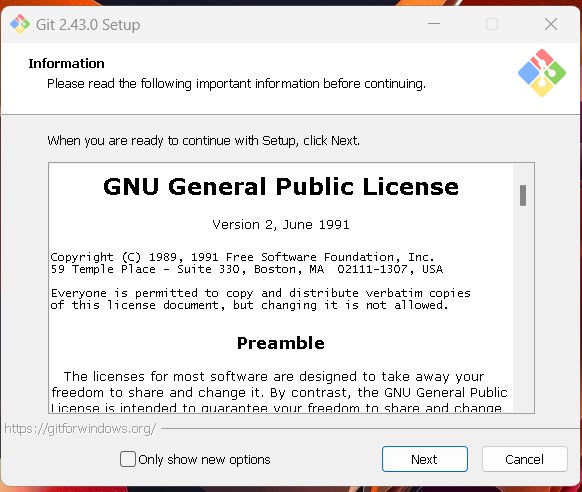
**How to install GIT on Windows?**

We can install Git on Windows, using the most official build which is available for download on the GIT’s official website or by just typing (scm git) on any search engine. We can go on  [https://git-scm.com/download/win a](https://git-scm.com/download/win)nd can select the platform and bit-version to download. And after clicking on your desired bit- version or ios it will start downloading automatically.

**Snapshots of download:**



*Figure 2 - Opted for “64-bit Git for Windows Setup*



*Figure 3*

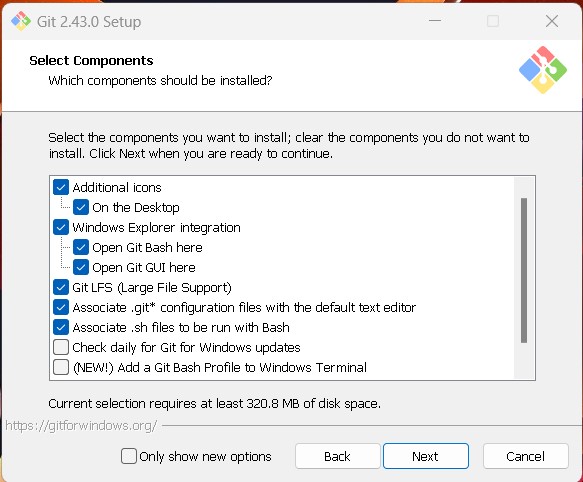
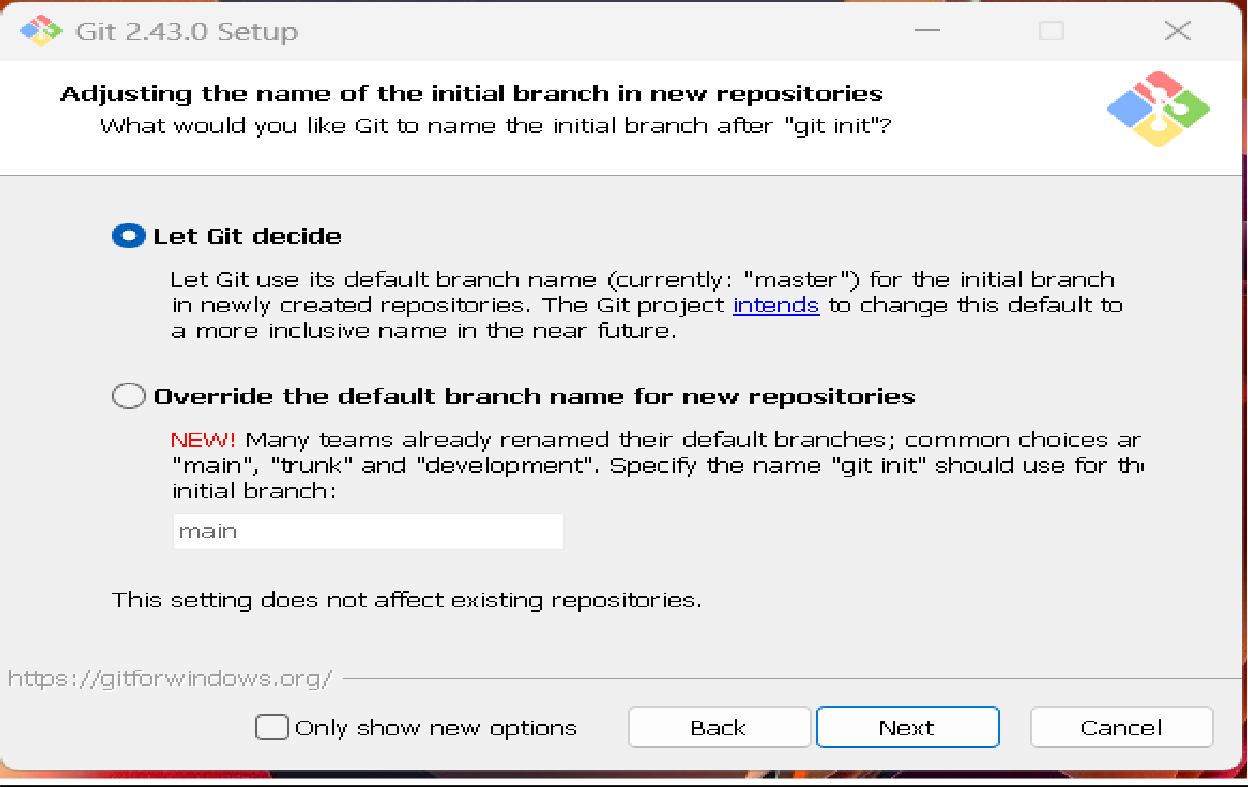
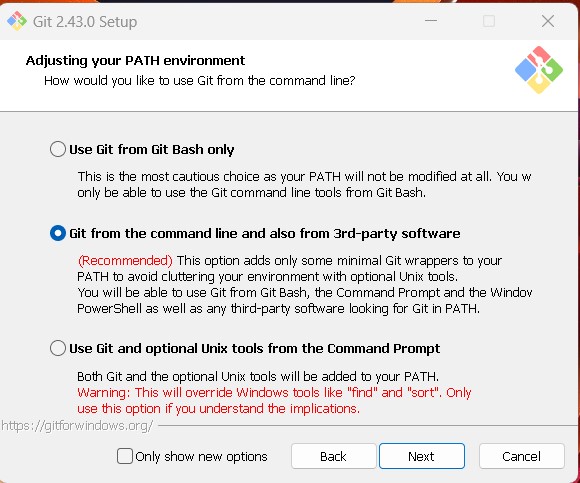


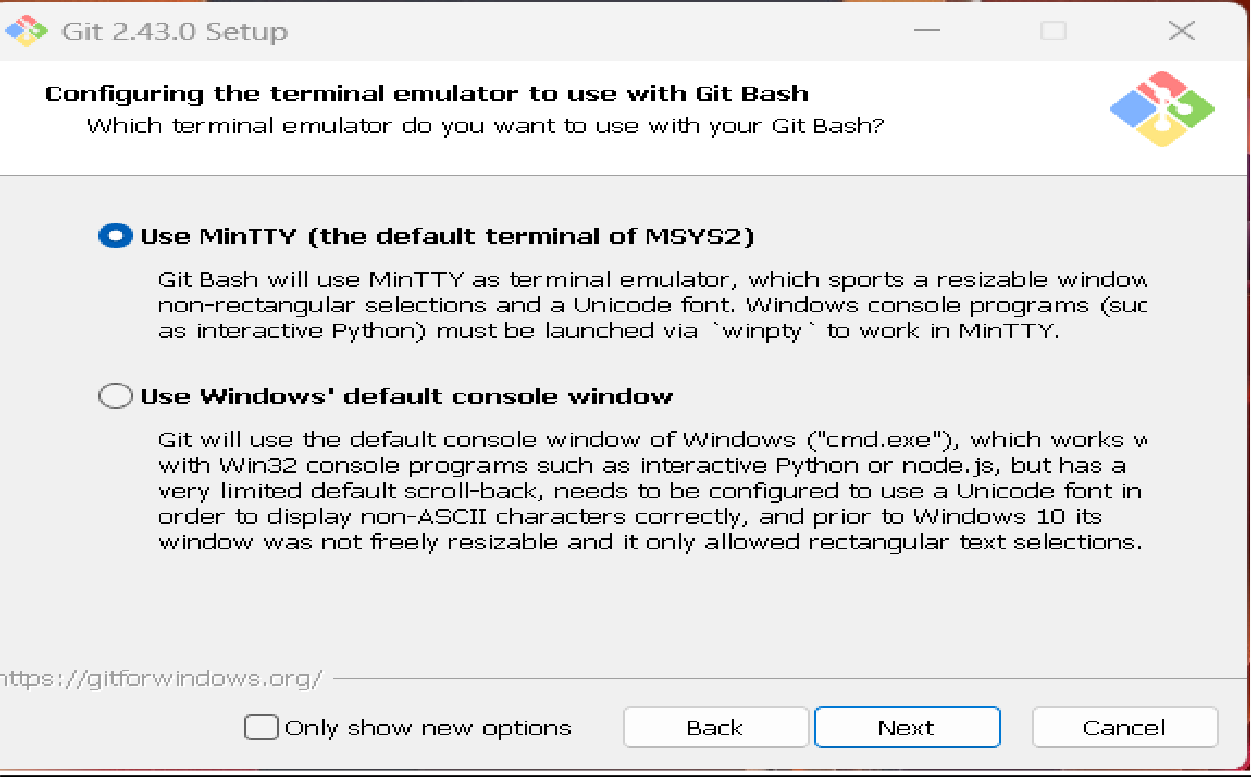
Figure 4 - Git and the selection of its components in downloads



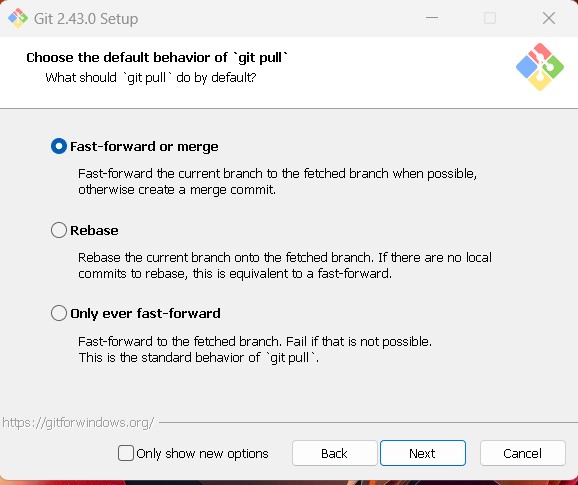
*Figure 5 - Selecting the git default branch*



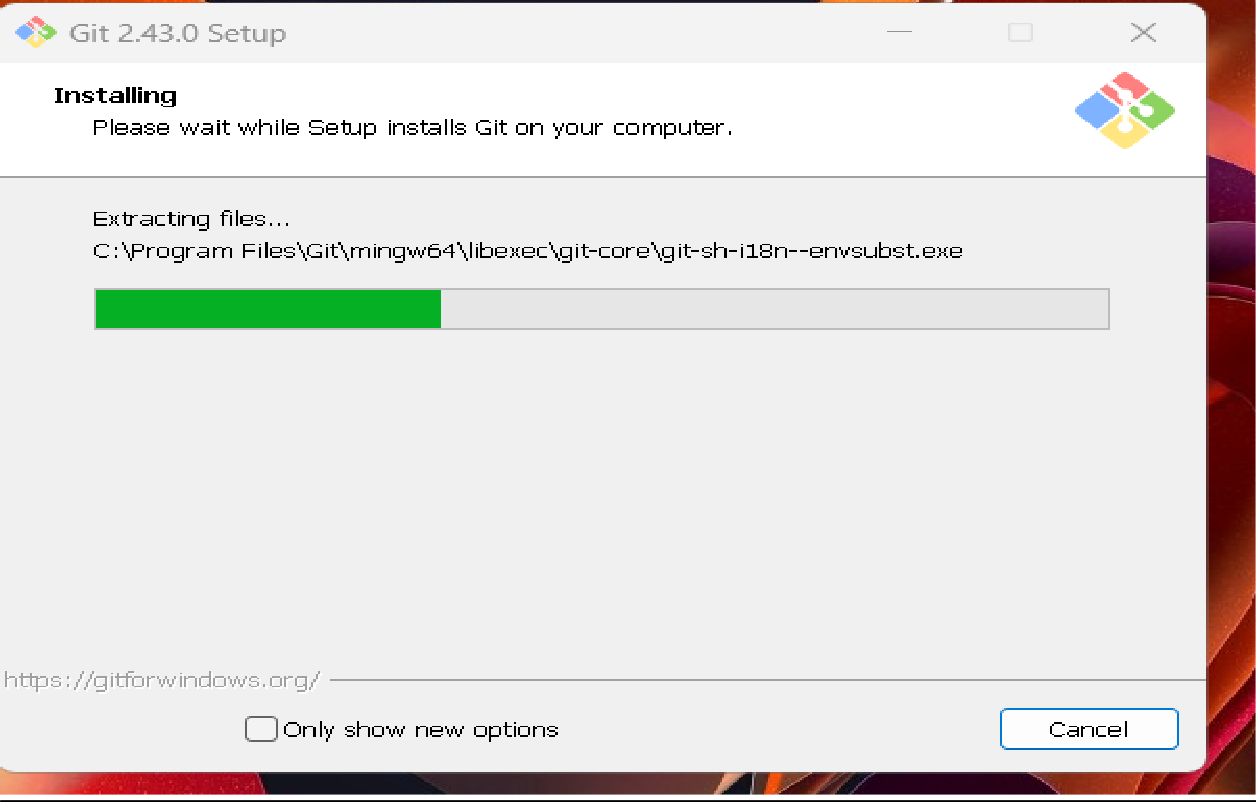
*Figure 6 - Choosing path for GitSetup*



*Figure 7 - Setting up the Terminal for git bash*



*Figure 8 - Choosing the git behaviour*

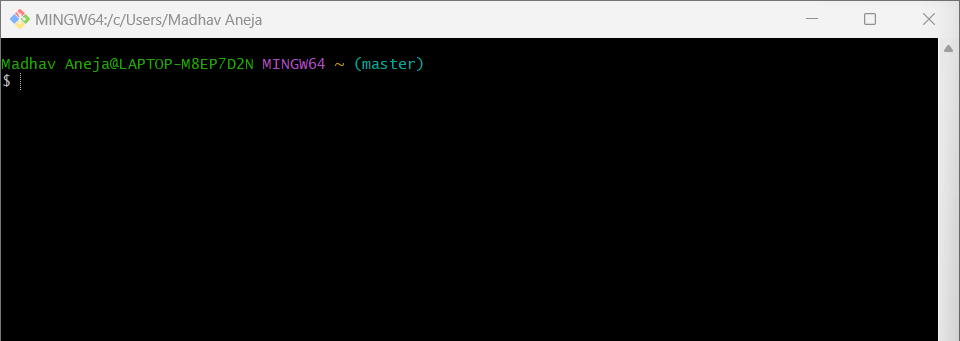


*Figure 9- Installing the setup of git*



*Figure 10 – Git setup completed*

Follow the installation instructions. After successful installation, click on the “Finish” button, it’ll automatically open Git Bash on your system, as shown in fig 1.3. Hence, this way git has been installed on your system.



*Figure 11 – Git Bah Launched*

Next, setup your account name and email by writing the commands, in your git bash window.

For this, we use command:

“git config --global user.name Name” “git config --global user.email email”

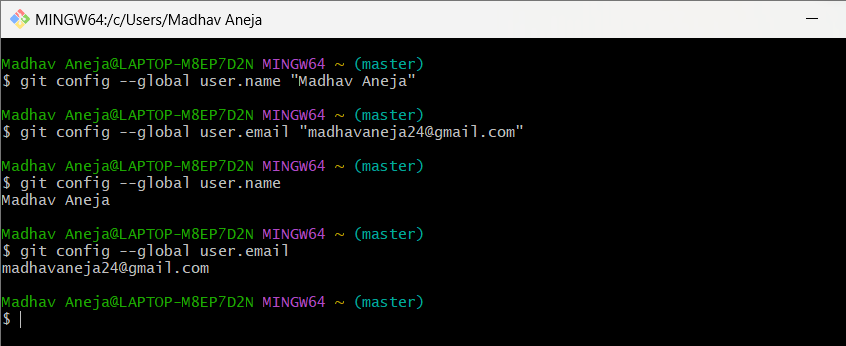
For verifying the user’s name and email, we use:

“git config --global user. name”

“git config --global user.email”

*Figure*

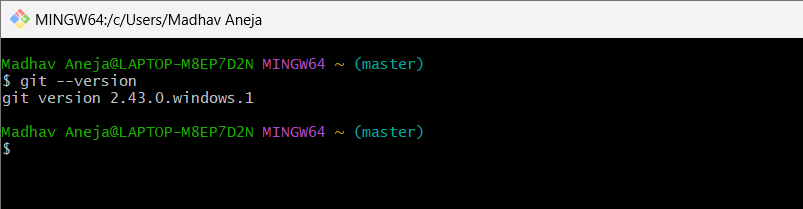
*12*



And, to verify that git is installed, run the following command,

*Figure*

*13*



# Experiment 2

## **Aim:** Setting up a GitHub Account

**Theory:**

**GitHub:** GitHub is a website and cloud-based service (client) that helps an individual or developers store and manage their code. We can also track as well as control changes to our or public code.

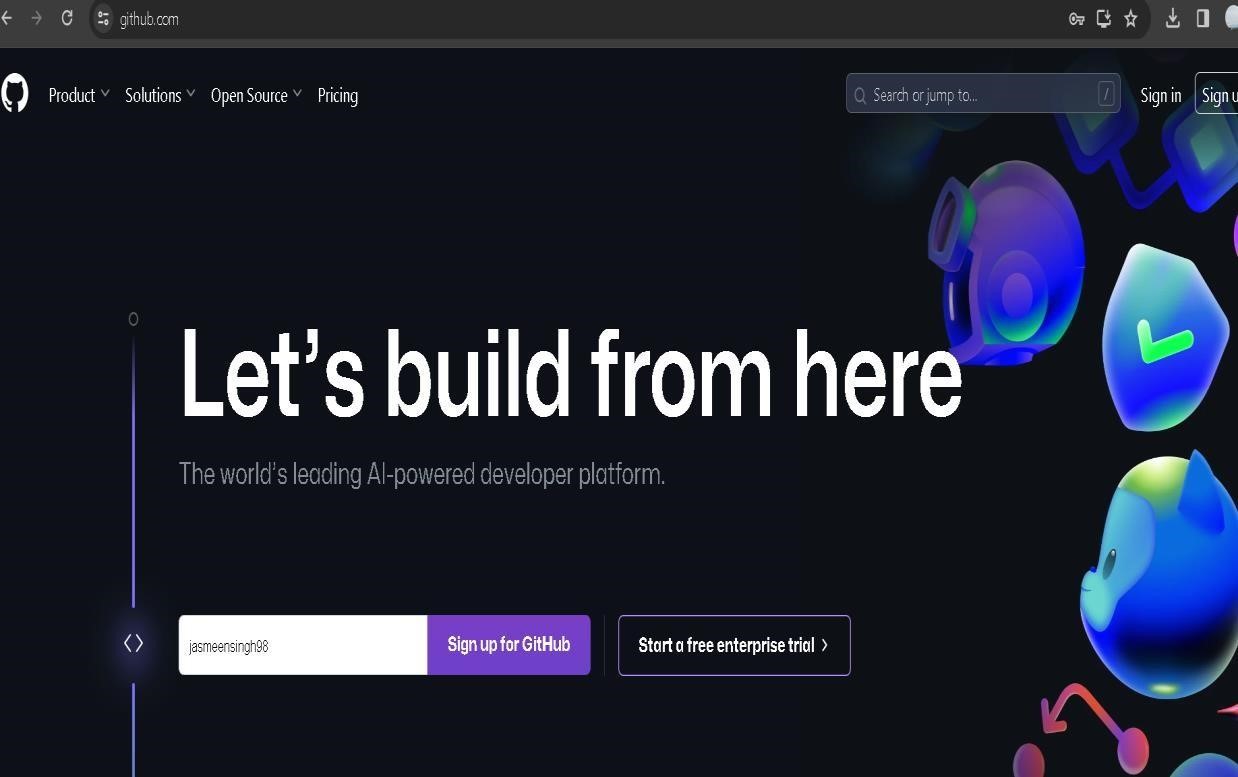
**Advantages of GitHub:**

GitHub has a user-friendly interface and is easy to use. We can connect the git-hub and git but using some commands shown below in figure 001. Without GitHub, we cannot use Git because it generally requires a host and if we are working on a project, we need to share it will our team members, which can only be done by making a repository. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

**Procedure:**

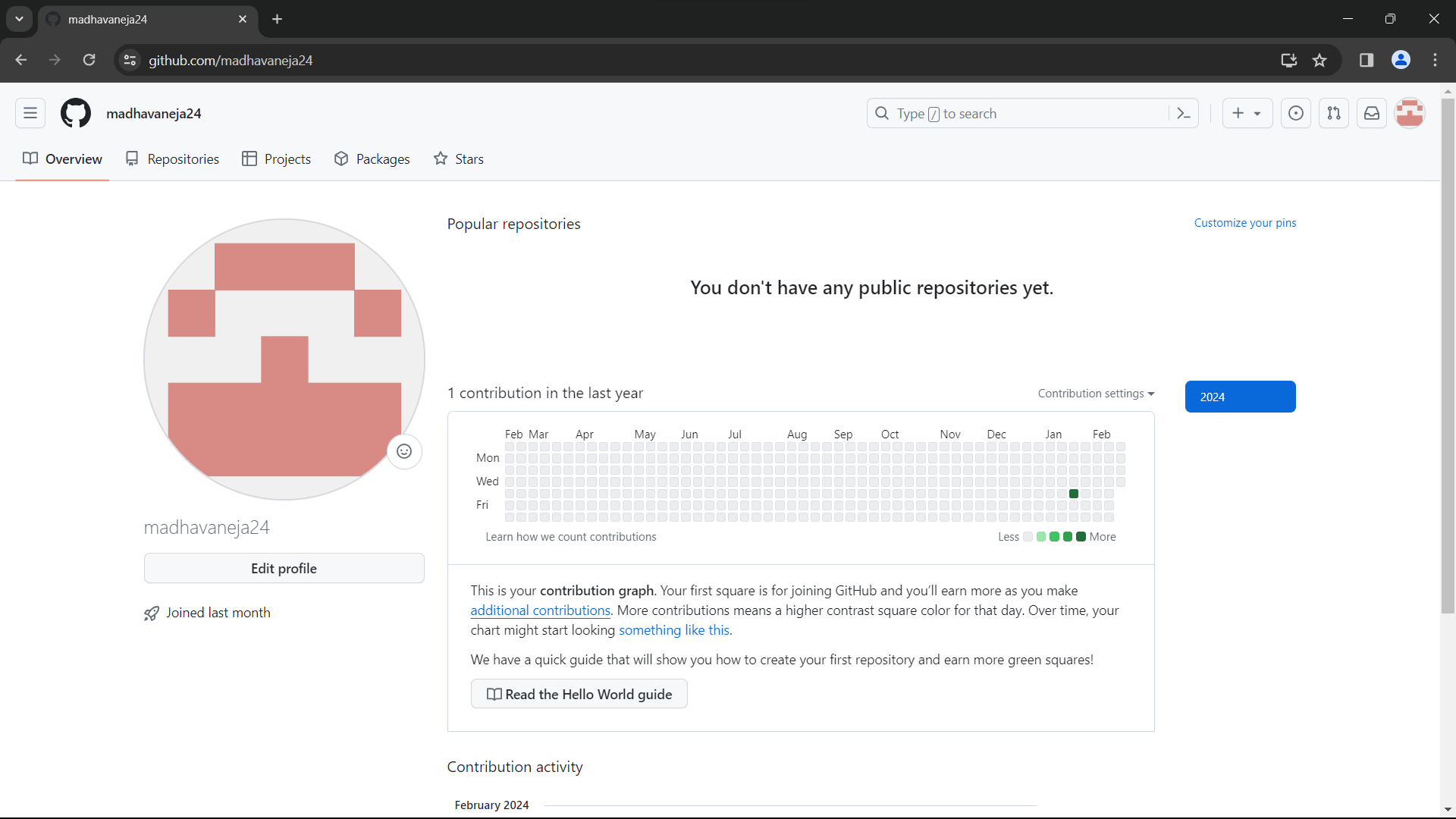
To make an account on GitHub, we search for GitHub on our browser or visit [https://github.com/signup.](https://github.com/signup) Then, we will enter our mail ID and create a username and password for a GitHub account.

**Snapshots:**

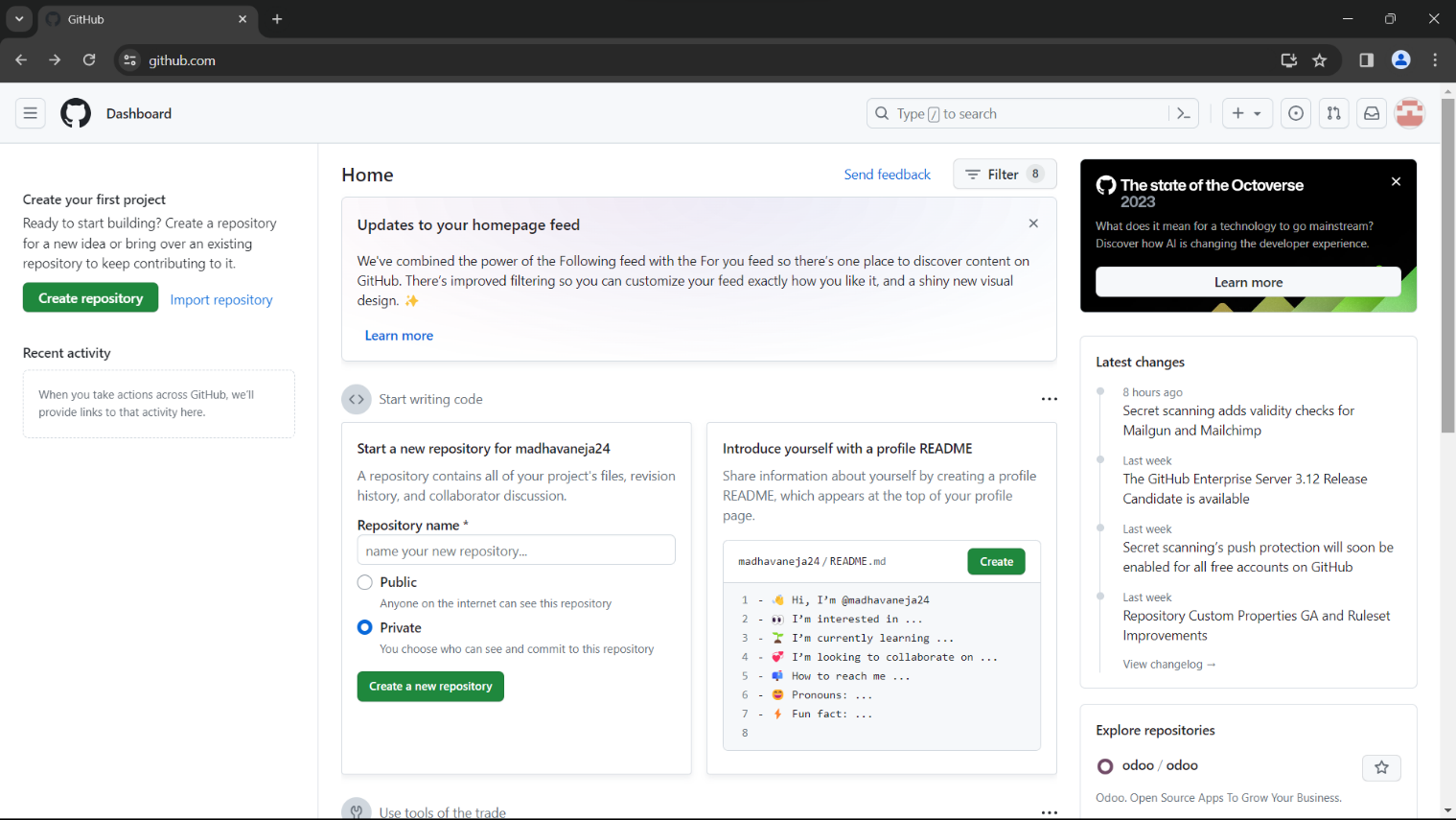


*Figure 14 – Creating account on GitHub*

After visiting the link this type of interface will appear, if you already have an account, you can sign in and if not, you can create.



*Figure 15 – GitHub Profile Page*



*Figure 16- GitHub Dashboard*

# Experiment 3

## **Aim:** Program to Generate log

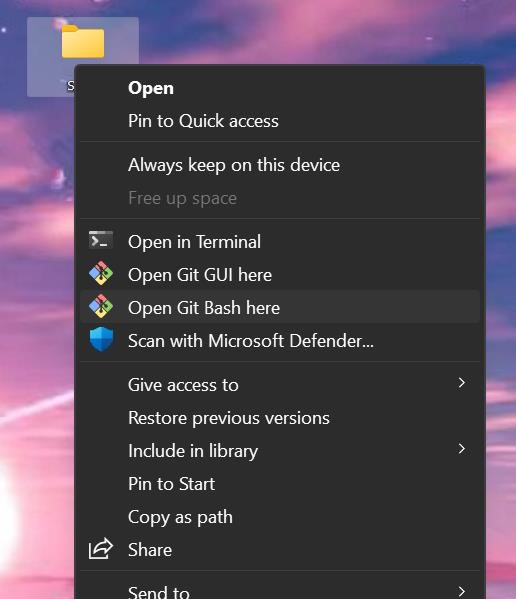
**Theory:**

Logs: Logs are nothing but the history which we can see in git by using the code git log.It contains all the past commits, insertions and deletions in it which we can see any time. Logs helps to check that what were the changes in the code or any other file and by whom. It also contains the number of insertions and deletions including at which time it was changed.

**Procedure:**

To generate a git log, we first need to make a git repository the steps to make soar as follows,

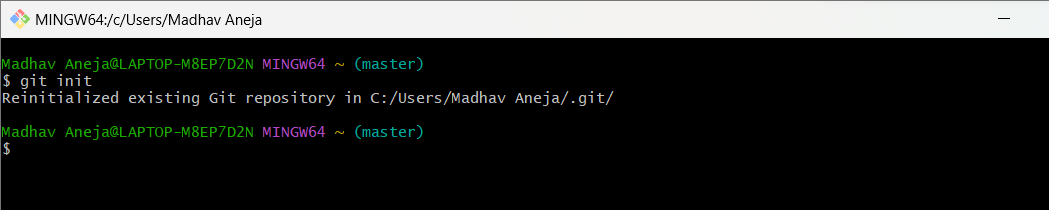
* Create a new folder in your local drive and right click on “open git bash here” this will open the git bash terminal.



*Figure 18*

* To create a new local repository, use command, “git init”, this will initialize git i.e.

create a hidden folder, “.git”.

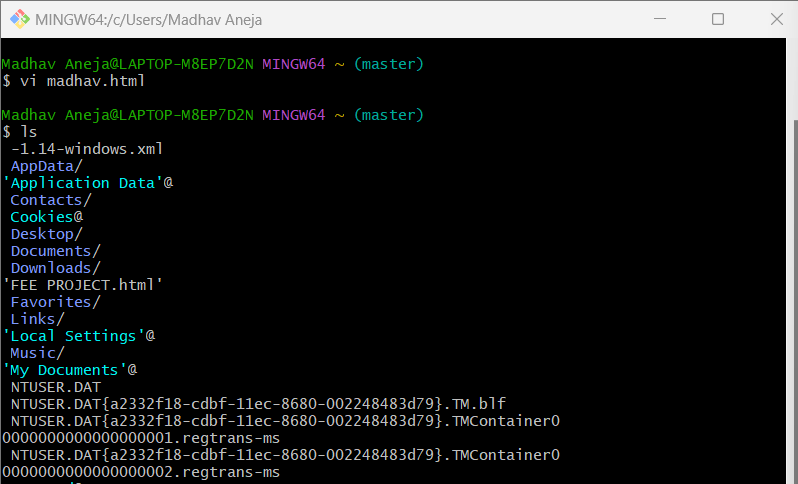


*Figure 19- Git init Command*

* Since, your username and email have already been configured with git, globally i.e. for all the files on the working system, we will not do it again.
* Next, we’ll create a file with the command, “vi <filename>” or by command “touch <file name>”, the file can be of any type i.e., .c, .txt, .py, etc.
* Then we’ll use the command, “ls”, it will list all the contents of that working directory, i.e

. all files and folders present in it.

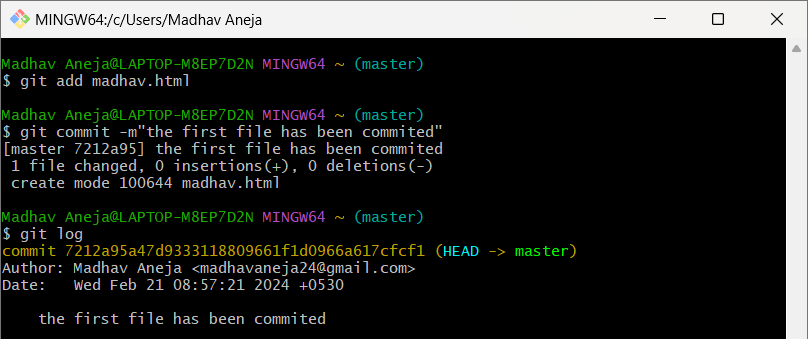
* After so, we’ll use the command “git status”, to display the state of the working directory and files.

c

*Figure 20- Git Command(s) interface - I*

* Now we will add our file to staging area for the next commit by using the command, “git add <filename>.

* Next, we’ll commit the changes made to our file by using the command, “gitcommit -m “message for commit”.
* Now, we’ll finally use the “git log” command to display the commit historyof the repository.



*Figure 21 Git Command(s) interface - II*

Similarly, you can create more files and commit them, after adding them to staging area and finally use the “git log” command to display the newer commit history along with previous one.

**Some Important Commands**

* **ls** It gives the file names in the folder.
* **ls -lart** Gives the hidden files also.
* **git status** Displays the state of the working directory and the staged snapshot.
* **touch filename** This command creates a new file in the repository.
* **clear** It clears the terminal.
* **rm -rf .git** It removes the repository.
* **git log** displays all of the commits in a repository's history
* **git diff** It compares my working tree to the staging area.

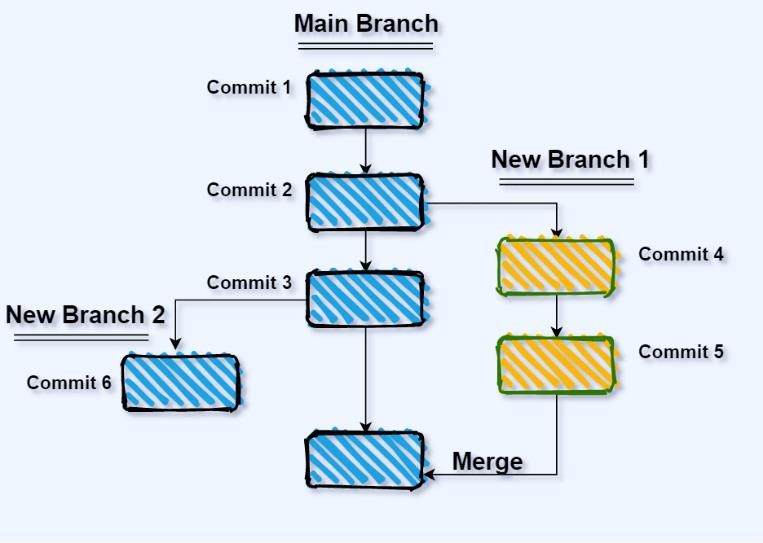
# Experiment 4

## **Aim:** Create and visualize branches

**Theory:**

**Branching**: A branch in Git is an independent line of work (a pointer to a specific commit). It allows users to create a branch from the original code (master branch) and isolate their work. Branches allow you to work on different parts of a project without impacting the main branch.

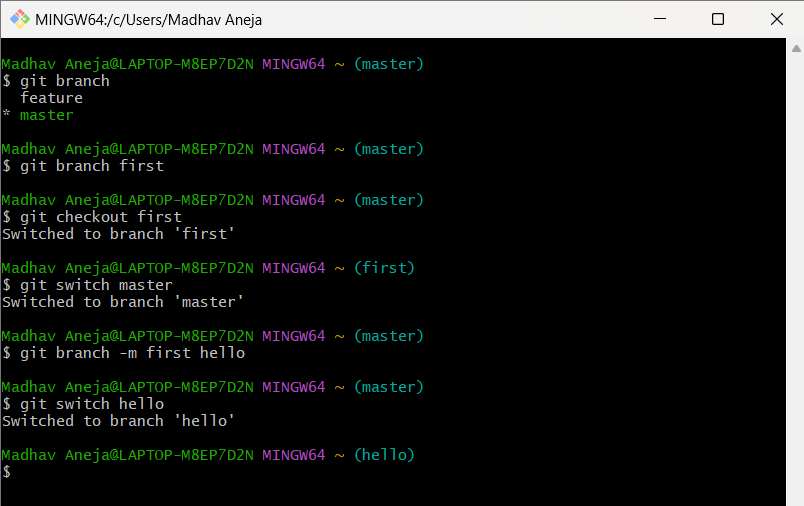
**Create branches**: The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child (other branches).

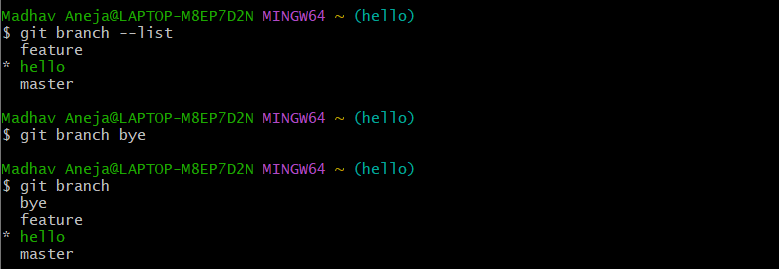


*Figure 22- Git Branching*

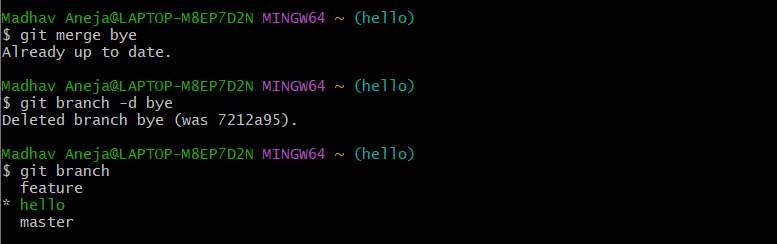
**Procedure:**

* **For creating a new branch:** git branch “name of the branch”
* **To check how many branches we have:** git branch or git branch --list
* **To change the present working branch:** git checkout “name of the branch” or git switch “branch name”
* **To rename a branch:** git branch -m “old name” “new name”
* The “\*” represents the current branch in which we are working.

 *Figure 23- Git Command(s) interface - I*



*Figure 24- Git Command(s) interface – II*

* **To merge a branch into another one:** git merge”branch name”
* **To delete a branch:** git branch -d”branch name”

*Figure 25- Git Command(s) interface - III*

# Experiment-5

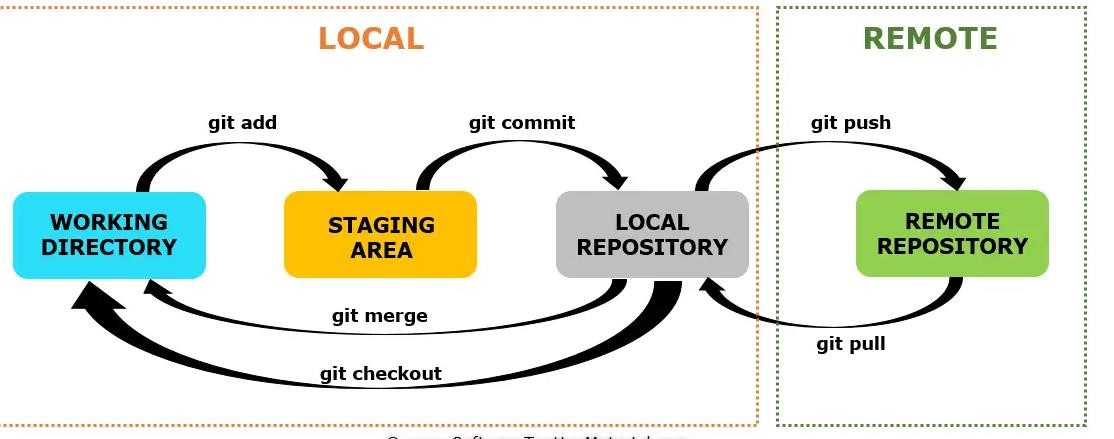
**Aim:** Git lifecycle description

**Theory:**

The Git lifecycle description refers to the lifecycle of changes within a Git repository, from creation to incorporation into the main branch. It outlines the typical flow of changes in a Git repository, from initial development to collaboration and integration with other tools and processes. It highlights the key stages and actions involved in managing and sharing code using Git. The three main states, where the files can reside in are, modified, staged and committed.

* Modified means that you have changed the file but have not committed it to your database yet.
* Staged means that you have marked a modified file in its current version to go into your next commit.
* Committed means that the data is safely stored in your local database.

This leads us to the three main sections of git lifecycle: the working tree, the staging area and the local repository. There’s another section called the remote repository.



*Figure 26- Git Lifecycle*

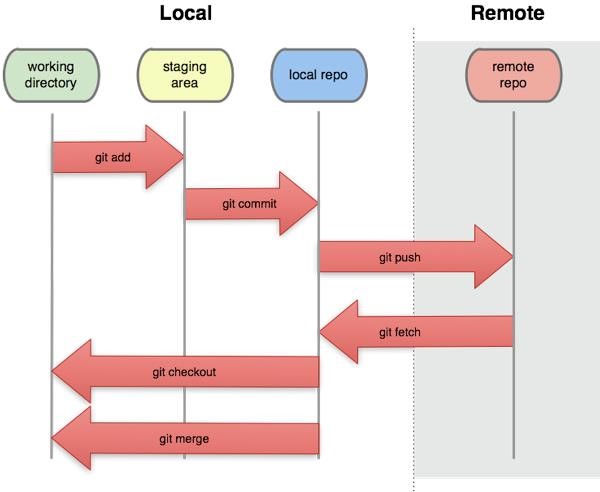
**Working Directory:** This is the directory on your local machine where you make changes to files in your Git repository. It's the initial stage in the Git lifecycle. Here, you edit files, add new ones, or delete existing ones. Overall, the working directory is where you actively develop and manage your project's files using Git. It serves as the interface between your local development environment and the version control system provided by Git.

**Staging Area:** The staging area, also known as the index, is where you prepare changes to be committed to the repository. When you're satisfied with the changes you've made in your working directory and want to include them in the next commit, you use the “git add” command to stage those changes. It provides a flexible and powerful mechanism for managing changes before committing them to the repository. It allows you to review, organize, and prepare changes for inclusion in commits, facilitating a structured and efficient development workflow.

**Local Repository:** The local repository is where Git stores committed changes.

When you use the “git commit -m <message>” command, Git takes the changes you've staged in the index and creates a new commit with them. This commit is then saved in your local repository along with a commit message and metadata. Changes made in the local repository are private to your local machine until you explicitly push them to a remote repository.

**Remote Repository:** The remote repository is a shared repository hosted on a remote server, such as GitHub, GitLab, or Bitbucket. After committing changes to your local repository, you may want to share those changes with others. You do this by pushing your commits to the remote repository. Remote repositories can have multiple contributors, allowing team members to collaborate on projects, review each other's code, and track changes over time.



*Figure 27- Major Git Sections*