Subject Name: **Source Code Management**

Subject Code: **22CS003**

Session: **2022-23**

Department: **DCSE**

Submitted By:
Vishal kumar
2210992536
G-18 – first year
Group 1

Submitted To:
DR. Simarjeet
Kaur

List of Programs

EXPERIMENT -01

Aim:

Setting up git client

Theory:

There are several ways to install Git on . The easiest is probably to install the Xcode Command Line Tools. If you don't have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A Git installer is maintained and available for download at the Git website, at If you don't have it installed already, it will prompt you to install it.

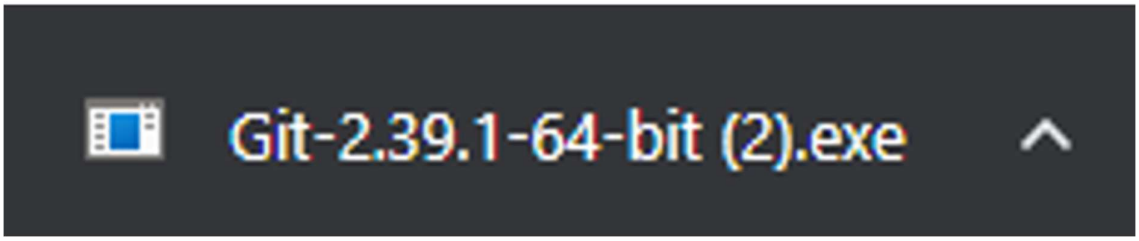
Procedure:

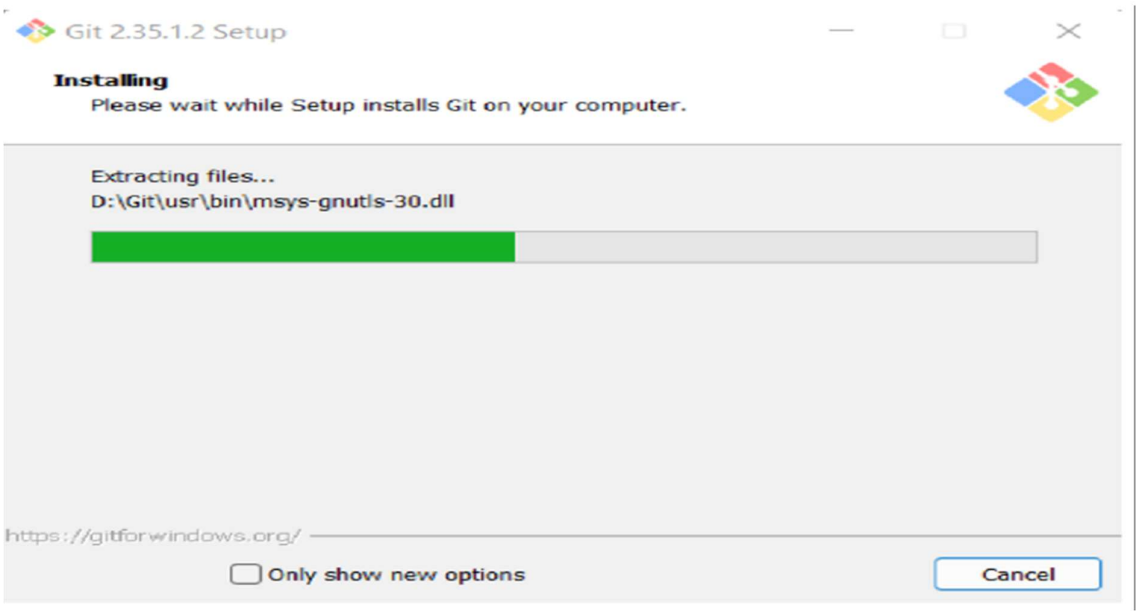
1)go to git scm.com

2)click download for windows

3)click on 64 bit for window setup

4) run.exe file

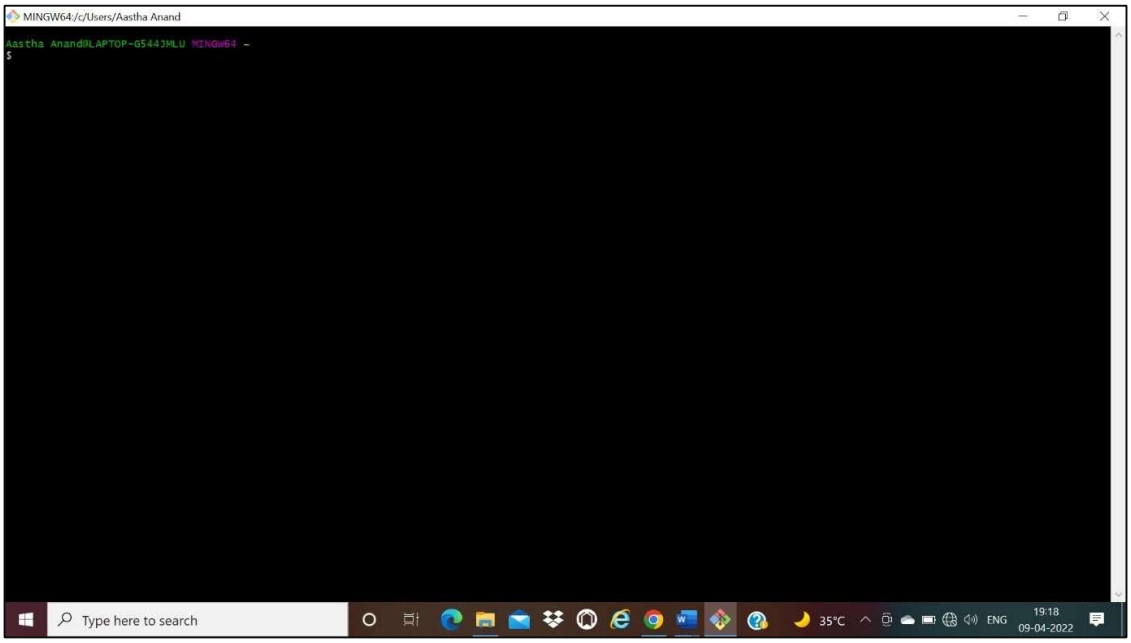




5)installing git

Name	Date modified	Type	Size
Git Bash	16-03-2022 08:51	Shortcut	2 KB
Git CMD	16-03-2022 08:51	Shortcut	2 KB
Git FAQs (Frequently Asked Questions)	16-03-2022 08:51	Internet Shortcut	1 KB
Git GUI	16-03-2022 08:51	Shortcut	2 KB
Git Release Notes	16-03-2022 08:51	Shortcut	2 KB

Git and its files in Downloads



Experiment -02

Aim:

Setting up a git hub account

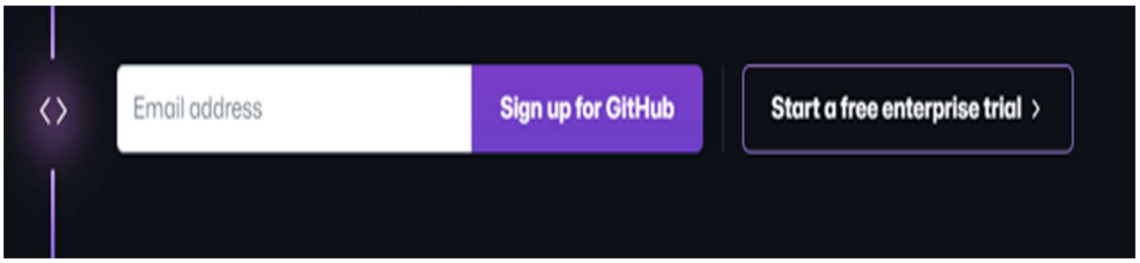
Theory :

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

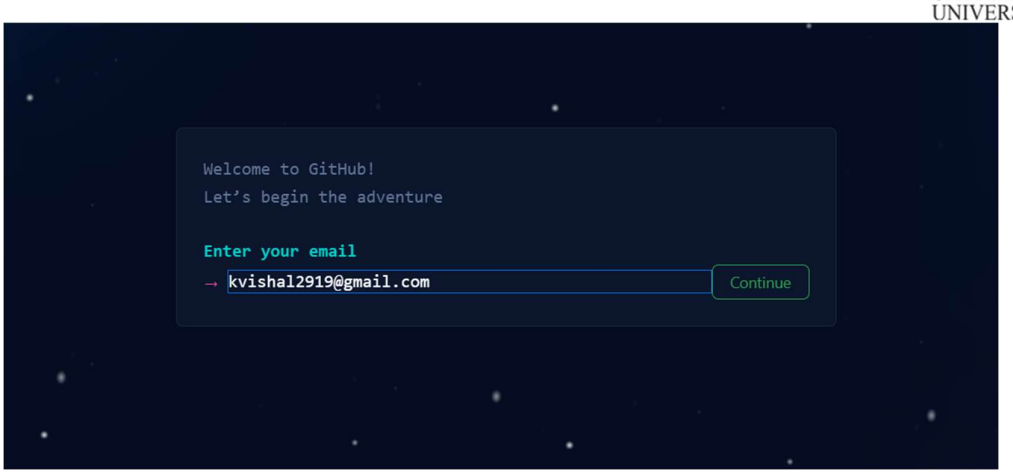
Advantages of GitHub: GitHub has a user-friendly interface and is easy to use.We can connect the git-hub and git but using some commands shown below in figure 01. Without GitHub we cannot use Git because it generally requires a host and if we are working for a project, we need to share it will our team members, which can only be done by making a repository.

Procedure:

1)go to hit hub.com

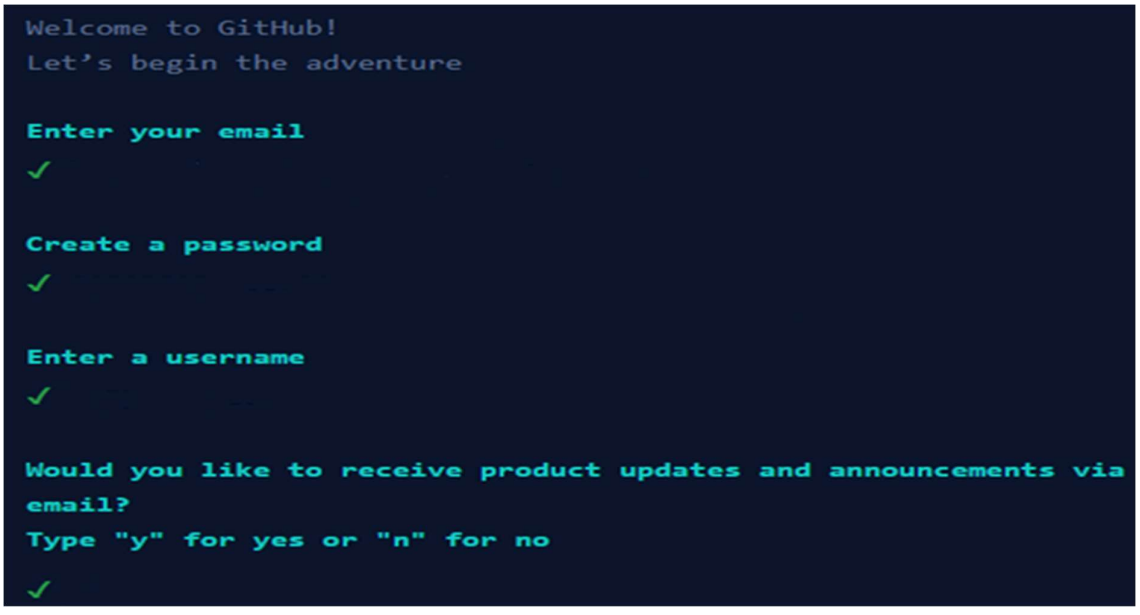


2)enter your email address

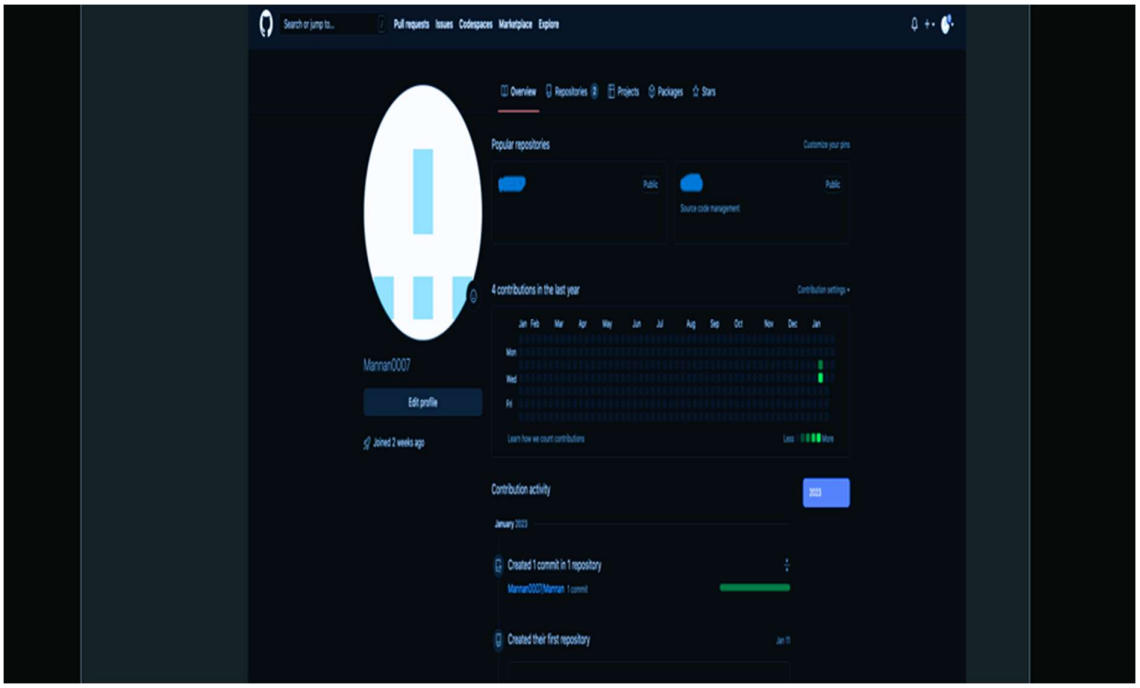


3)create a password

4)Setting up your account



5)your account has been setup



Source code management (SCM) is used to track modifications to a source code repository. SCM tracks a running history of changes to a code base and helps resolve conflicts when merging updates from multiple contributors.

EXPERIMENT -03

Aim:

Generate git log

THEORY:

git log : it is used to view repository and all the changes are also visible that we have made.it also tell us the name of author and time.

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:

List commits that are reachable by following the `parent` links from the given commit(s), but exclude commits that are reachable from the one(s) given with a ^ in front of them. The output is given in reverse chronological order by default.

```
MINGW64:/c/Users/91754/vishal23
$ cd vishal23

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git init
Initialized empty Git repository in C:/Users/91754/vishal23/.git/

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ ls

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ ls -a
./  ../  .git/

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ cat > a.txt

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ cat > b.txt

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git add .

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git commit -m "2 files added"
[master (root-commit) ae23032] 2 files added
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
 create mode 100644 b.txt

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git log
commit ae230327e5f4532abe03821c6548cb274a92e3ce (HEAD -> master)
Author: vishal kumar <vishal@1234>
Date:   Thu Feb 9 11:07:07 2023 +0530

    2 files added
```

When we use GIT for the first time, we have to give the user name and email so that if I am going to change in project, it will be visible to all.

For this, we use command:

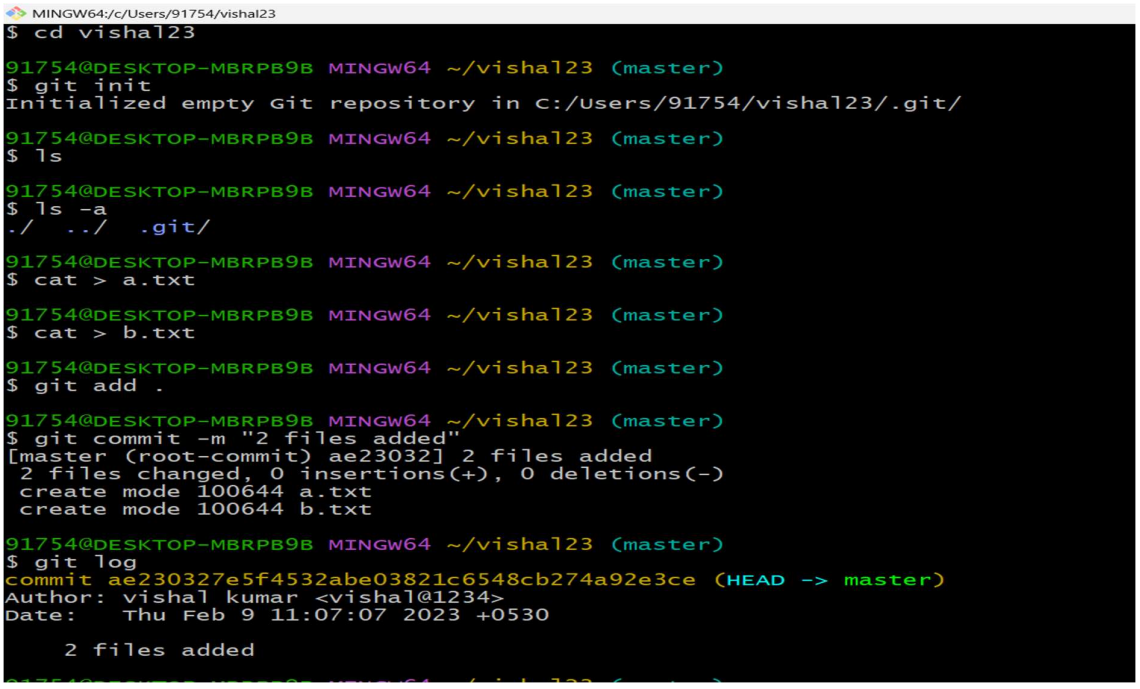
"git config --global user.name Name"
"git config --global user. email "email"

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

"git config --global user.name"
"git config --global user .email"

The git log command displays a record of the commits in a Git repository. By default, the git log command displays a commit hash, the commit message and other commit metadata

- 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 staging area.



Experiment -04

AIM:

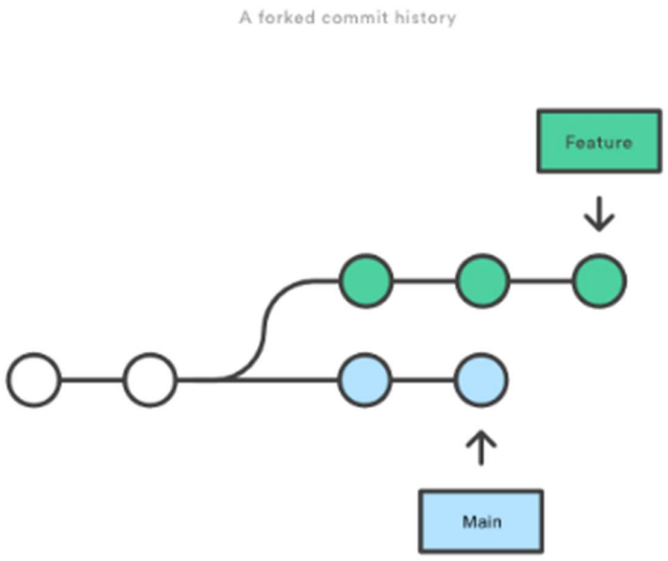
Create and visualize branches

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.

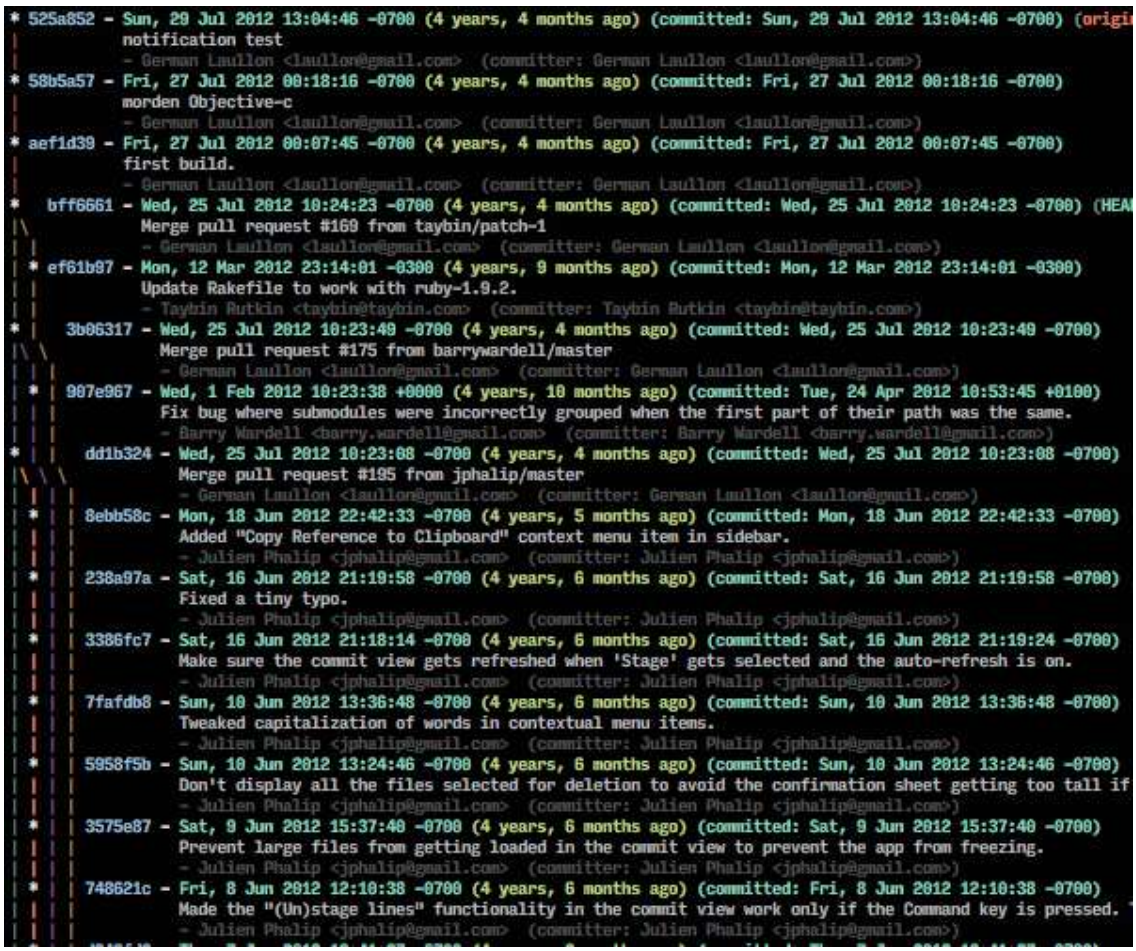
Create branches: The main branch in git is called as master branch. But we can make branches out of this main master branch. All the files present in master can be shown in branch but the file which are created in branch are not shown in master branch. We can also merge both the parent (master) and child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

For creating a new branch, git branch name by default is master branch



Procedure:

```
$ git log --oneline
7398d31 (HEAD -> master) Merge branch 'feature-branch'
257aee9 change content
95f2205 change content
4b85ba2 HTML restructering
fc01a24 (tag: v1.0) Add an empty rulset for a body selector
10c61dd add an h1 selector
bba41e4 changes from the reset
cafdc1f resolve the conflict
651acb8 restructure the html document (indentation)
e378f11 add some styling to the h1 element
9fab2b5 add html structure
74835de Inital commit
```



MINGW64:/c/Users/91754/vishal23

```
$ git commit -m "2 files added"
[master (root-commit) ae23032] 2 files added
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 a.txt
 create mode 100644 b.txt

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git log
commit ae230327e5f4532abe03821c6548cb274a92e3ce (HEAD -> master)
Author: vishal kumar <vishal@1234>
Date:   Thu Feb 9 11:07:07 2023 +0530

    2 files added

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git branches
git: 'branches' is not a git command. See 'git --help'.

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git branch
* master

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git branch oppo

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (master)
$ git checkout oppo
Switched to branch 'oppo'

91754@DESKTOP-MBRPB9B MINGW64 ~/vishal23 (oppo)
$ git branch
  master
* oppo
```

Experiment -05

Aim:

Git lifecycle description

Theory:

Stages in GIT Life Cycle: Files in a Git project have various stages like Creation, Modification, Refactoring, and Deletion and so on. Irrespective of whether this project is tracked by Git or not, these phases are still prevalent. However, when a project is under Git version control system, they are present in three major Git states in addition to these basic ones. Here are the three Git states:

- Working directory
- Staging area

Git directory

Procedure:

- Working directory :- Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

Staging area :- Staging area is the playground where you group, add and organize the files to be committed to Git for tracking their versions.

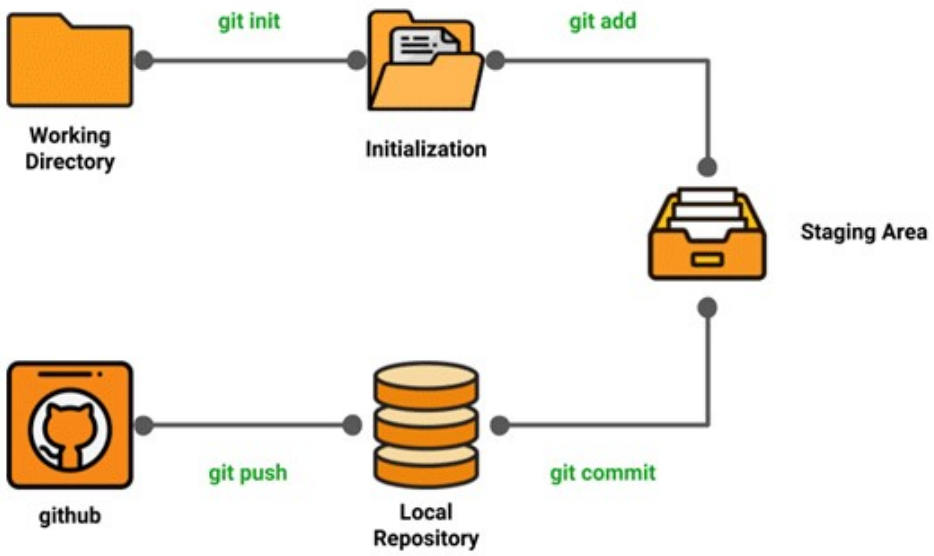
Git Directory:

Now that the files to be committed are grouped and ready in the staging area, we can commit these files. So, we commit this group of files along with a commit message explaining what is the commit about. Apart from commit message, this step also records the author and time of the commit. Now, a snapshot of the files in the commit is recorded by Git. The information related to this commit is stored in the Git directory.

Remote Repository: It means mirror or clone of the local Git repository in GitHub. And pushing means uploading the commits from local Git repository to remote repository hosted in GitHub.

- clone the Git repository as a working copy.
- modify the working copy by adding/editing files.
- If necessary, you also update the working copy by taking other developer's changes.
- review the changes before commit.
- commit changes. If everything is fine, then you push the changes to the repository.

- After committing, if you realize something is wrong, then you correct the last commit and push the changes to the repository.



- It's the git life-cycle