Source Code Management

File Subject Name: Source Code Management (SCM)

Subject Code: CS181

Submitted By:

Name: **Manav Taneja**

Roll No.**2310991975**

Group: **G22-B**

Task 1.1 Submission (Week 4)

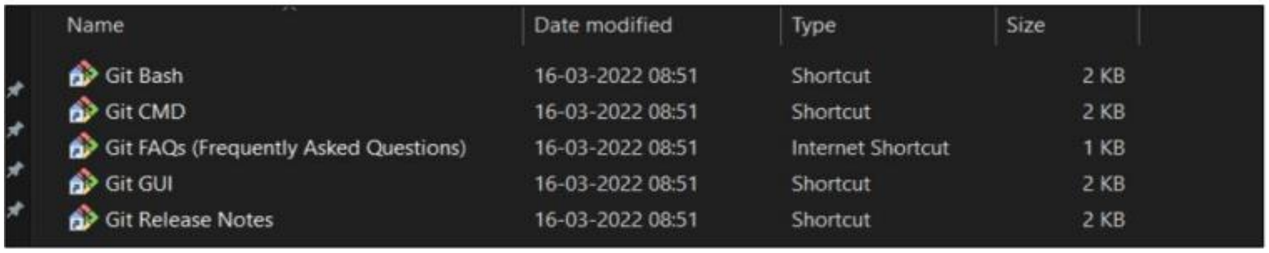
1. Setting up of Git Client
2. Setting up GitHub Account
3. Generate logs
4. Create and visualize branches
5. Git lifecycle description

EXPERIMENT-1

Aim: Setting up of Git Client

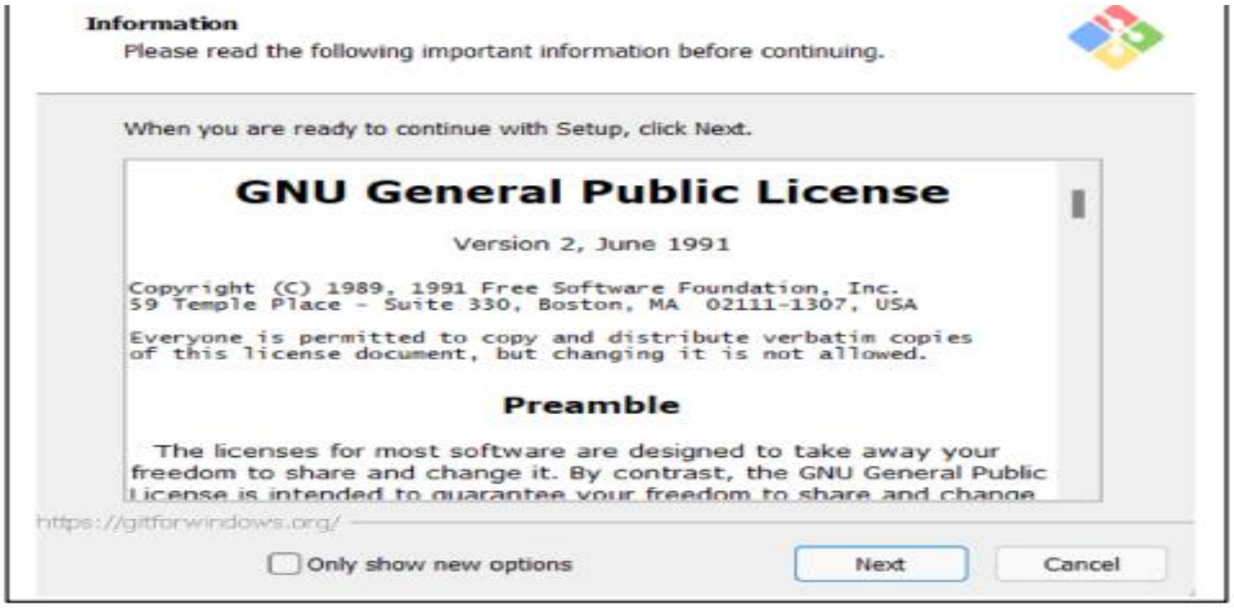
Theory: GIT: It's a Version Control System (VCS). It is a software or we can say a server by which we are able to track all the previous changes in the code. It is basically used for pushing and pulling of code. We can use git and git-hub parallelly to work with multiple members or individually. We can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

Procedure: 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 and can select the platform and bit-version to download. And after clicking on your desired bit-version or ios it will start downloading automatically.



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 download


Information

Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

GNU General Public License

Version 2, June 1991

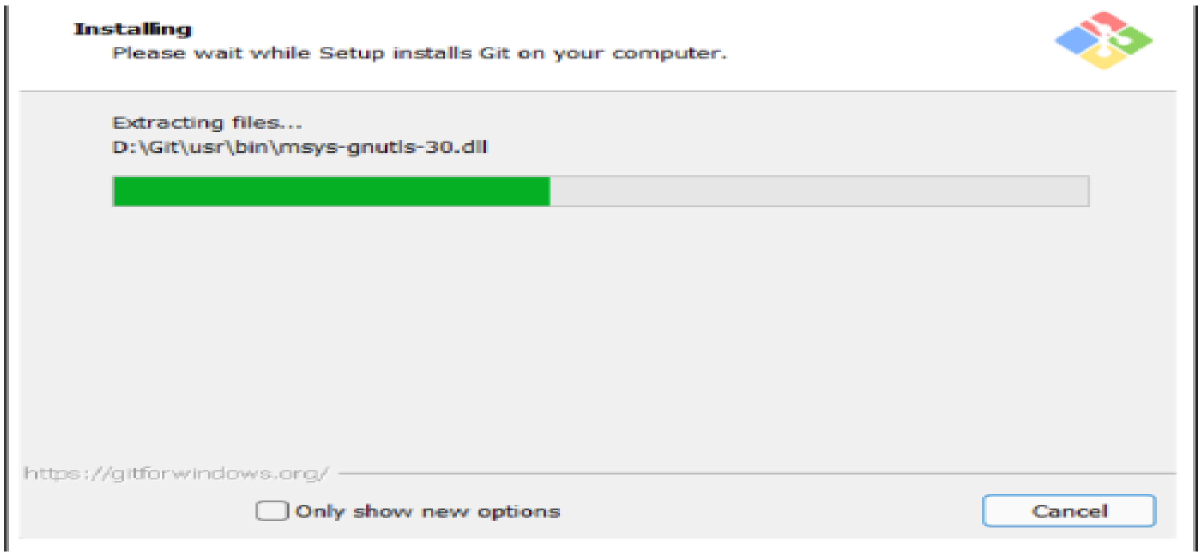
```
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA  02111-1307, USA

Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
```

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change

https://gitforwindows.org/

Git download and license

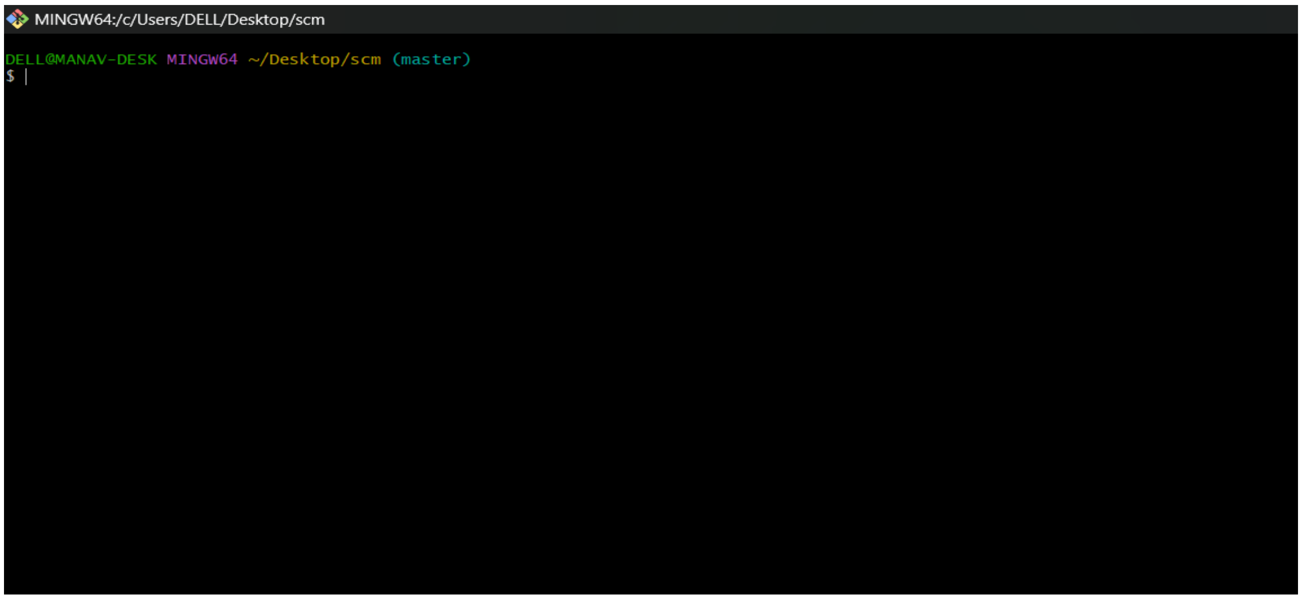

Installing

Please wait while Setup installs Git on your computer.

Extracting files...
D:\Git\usr\bin\msys-gnutls-30.dll

https://gitforwindows.org/

Installation



```
MINGW64:/c/Users/DELL/Desktop/scm

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ |
```

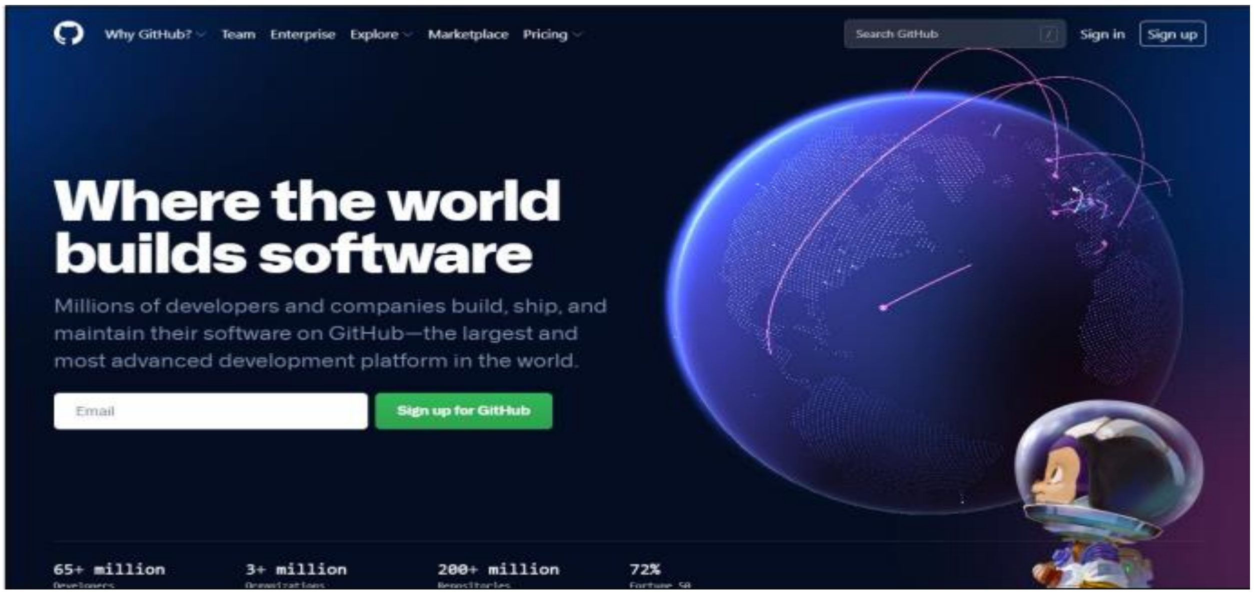
Git bash terminal

Experiment 2

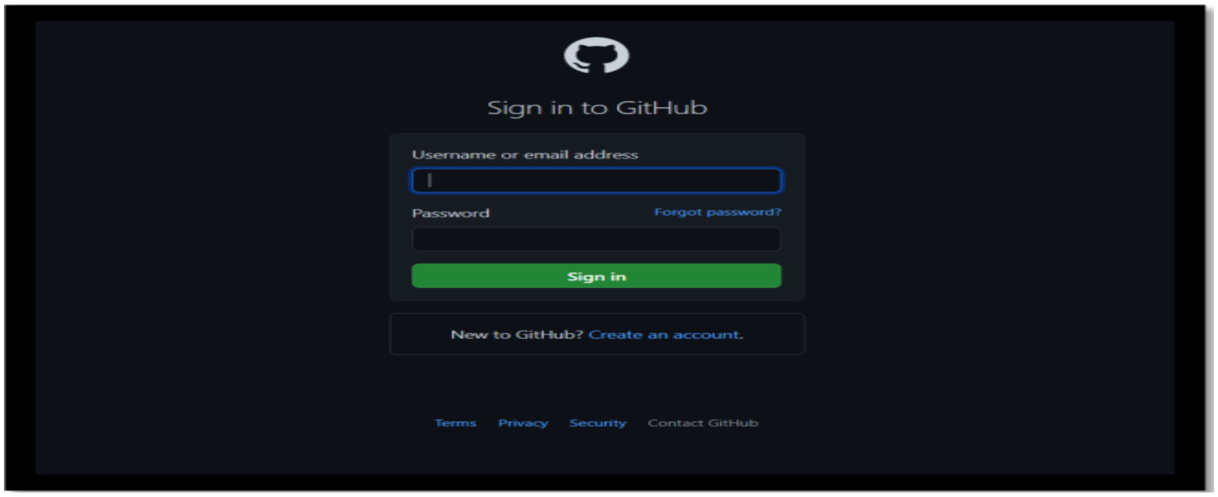
Aim: Setting up GitHub Account

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 001. 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. Additionally, anyone can sign up and host a public code repository for free, which makes GitHub especially popular with open-source projects.

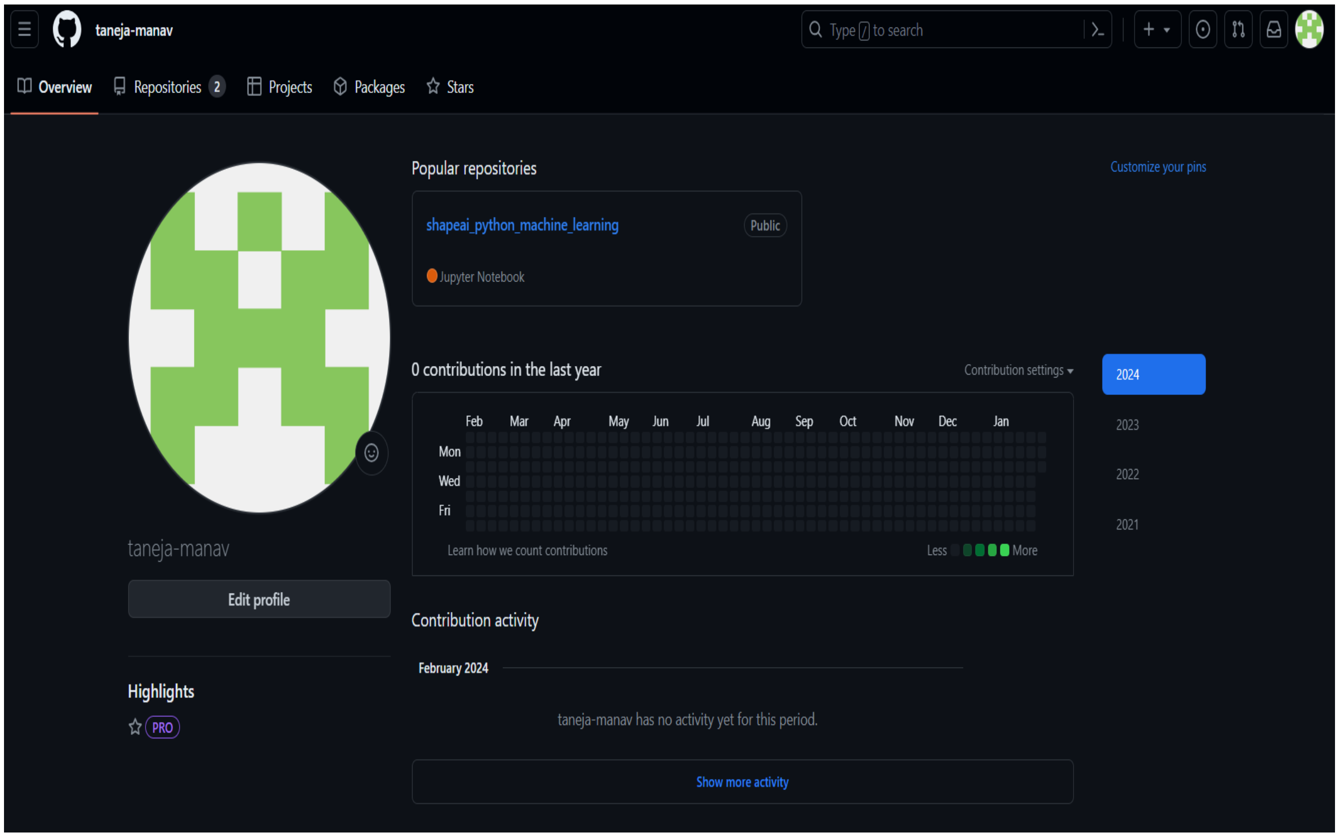
Procedure: To make an account on GitHub, we search for GitHub on our browser or visit https://github.com/signup. Then, we will enter our mail ID and create a username and password for a GitHub account


Why GitHub?
Team
Enterprise
Explore
Marketplace
Pricing
Search GitHub
Sign in
Sign up
Where the world builds software
Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.
Email
Sign up for GitHub
65+ million
3+ million
200+ million
72%

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.



Login



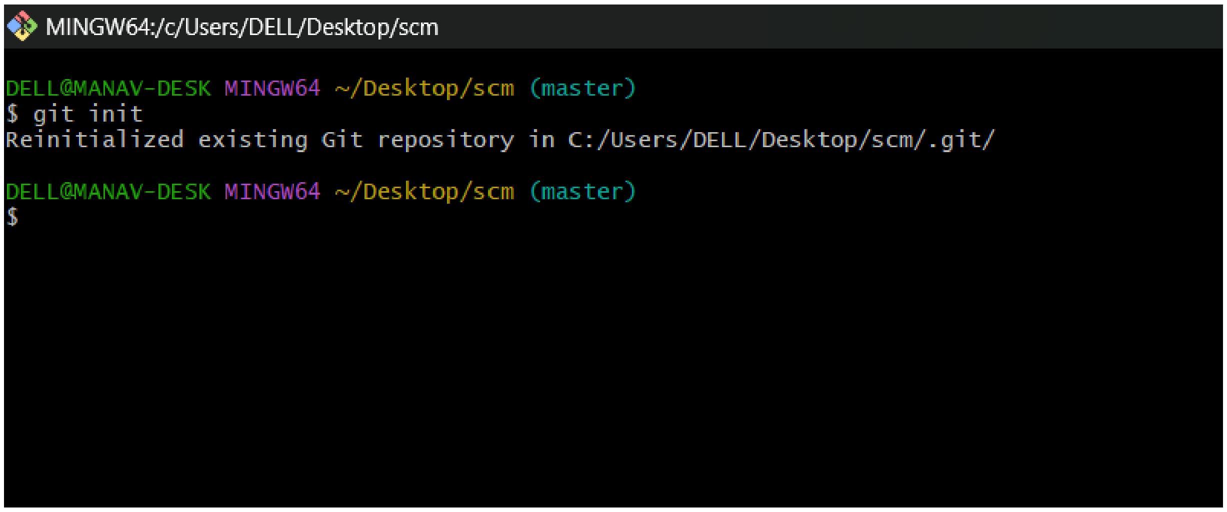
GitHub interface

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 at any time. Logs help 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: Firstly, create a local repository using Git. For this, you have to make a folder in your device, right click and select "Git Bash Here". This opens the Git terminal. To create a new local repository, use the command "git init" and it creates a folder ". git"



```
MINGW64:/c/Users/DELL/Desktop/scm

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git init
Reinitialized existing Git repository in C:/Users/DELL/Desktop/scm/.git/

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$
```

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”

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.

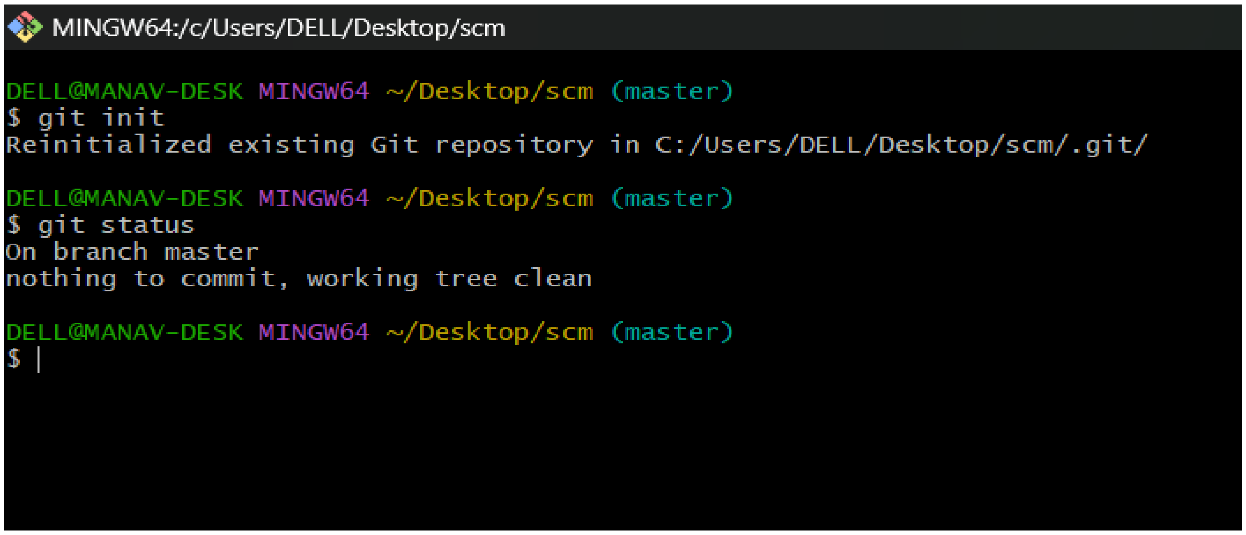
• touch filename → This command creates a new file in the repository

. • Clear → It clears the terminal.

• rm -rf .git → It removes the repository.

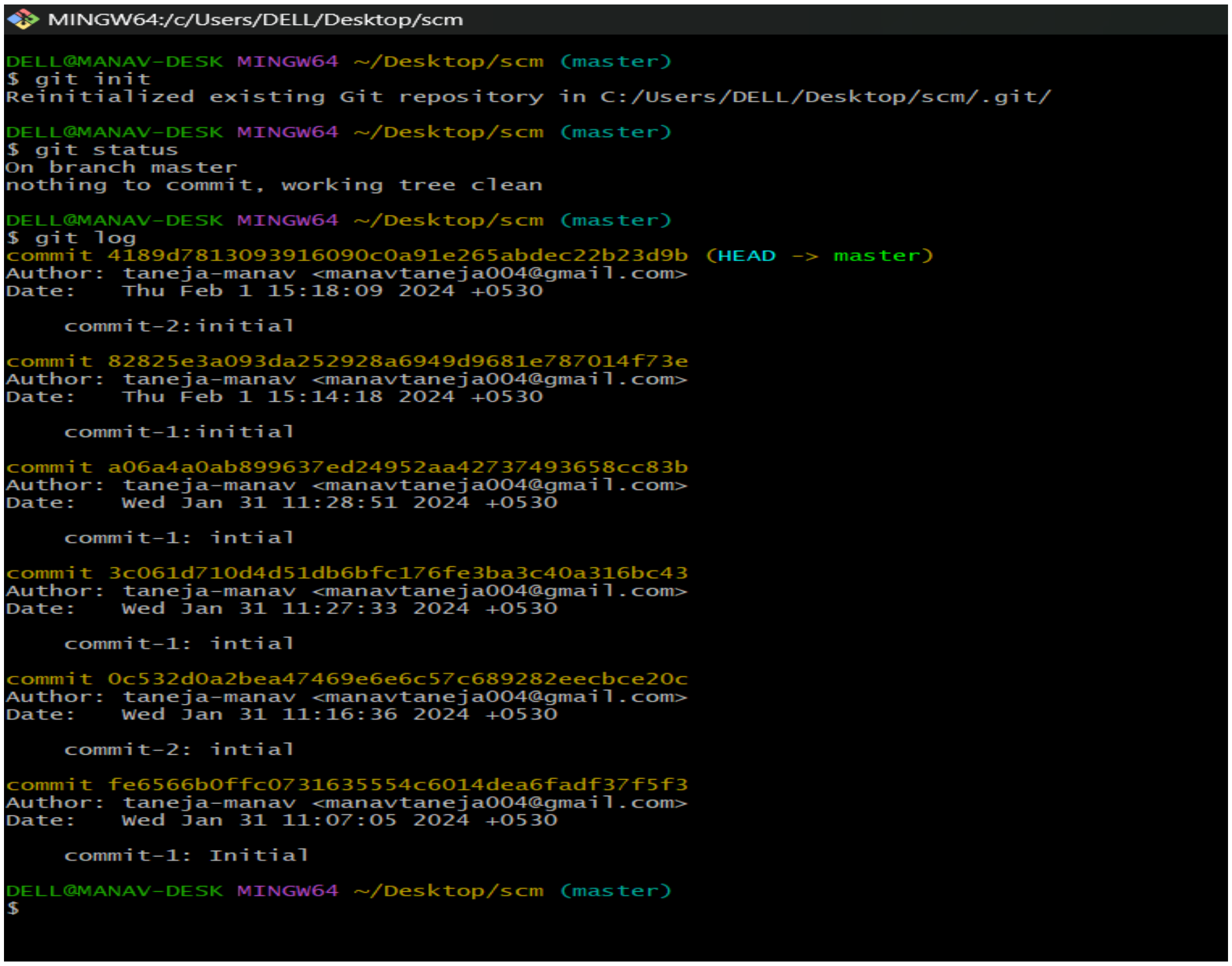
• git log → displays all of the commits in a repository's history

• git diff → It compares my working tree to staging area.



```
MINGW64:/c/Users/DELL/Desktop/scm

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git init
Reinitialized existing Git repository in C:/Users/DELL/Desktop/scm/.git/

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git status
On branch master
nothing to commit, working tree clean

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ |
```

Git status



```
MINGW64:/c/Users/DELL/Desktop/scm

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git init
Reinitialized existing Git repository in C:/Users/DELL/Desktop/scm/.git/

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git status
On branch master
nothing to commit, working tree clean

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git log
commit 4189d7813093916090c0a91e265abdec22b23d9b (HEAD -> master)
Author: taneja-manav <manavtaneja004@gmail.com>
Date:   Thu Feb 1 15:18:09 2024 +0530

    commit-2:initial

commit 82825e3a093da252928a6949d9681e787014f73e
Author: taneja-manav <manavtaneja004@gmail.com>
Date:   Thu Feb 1 15:14:18 2024 +0530

    commit-1:initial

commit a06a4a0ab899637ed24952aa42737493658cc83b
Author: taneja-manav <manavtaneja004@gmail.com>
Date:   Wed Jan 31 11:28:51 2024 +0530

    commit-1: intial

commit 3c061d710d4d51db6bfc176fe3ba3c40a316bc43
Author: taneja-manav <manavtaneja004@gmail.com>
Date:   Wed Jan 31 11:27:33 2024 +0530

    commit-1: intial

commit 0c532d0a2bea47469e6e6c57c689282eecbce20c
Author: taneja-manav <manavtaneja004@gmail.com>
Date:   Wed Jan 31 11:16:36 2024 +0530

    commit-2: intial

commit fe6566b0ffc0731635554c6014dea6fadf37f5f3
Author: taneja-manav <manavtaneja004@gmail.com>
Date:   Wed Jan 31 11:07:05 2024 +0530

    commit-1: Initial

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$
```

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.

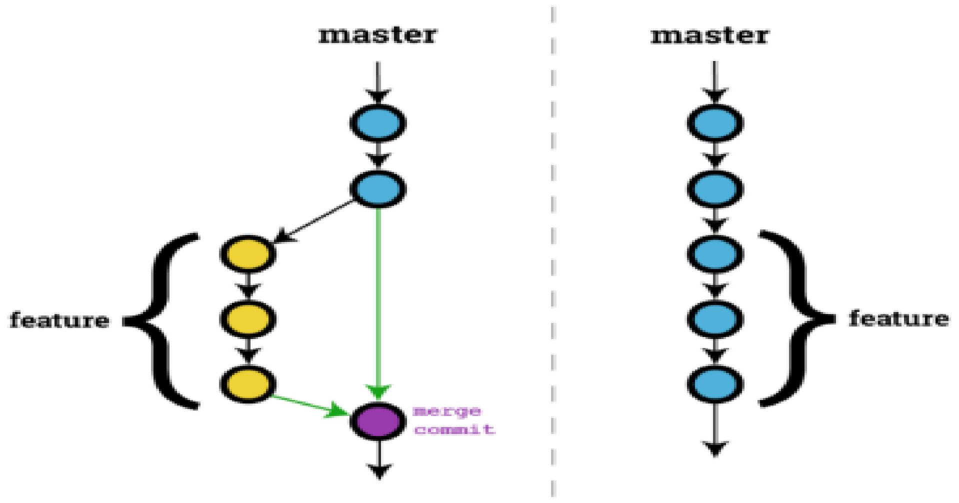
EXPERIMENT-4

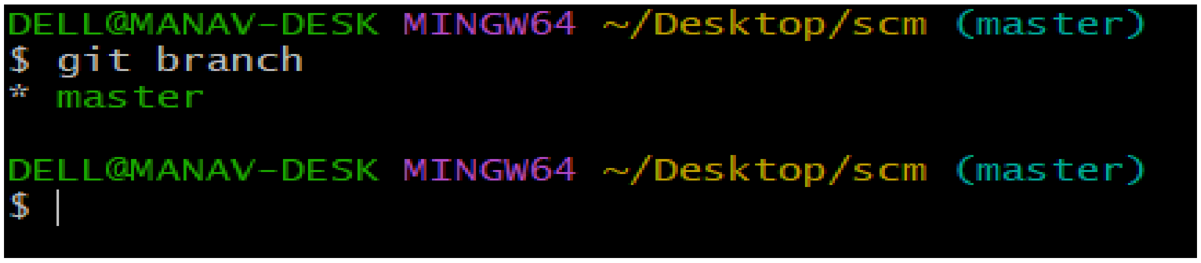
Aim: Create and visualize branches

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).

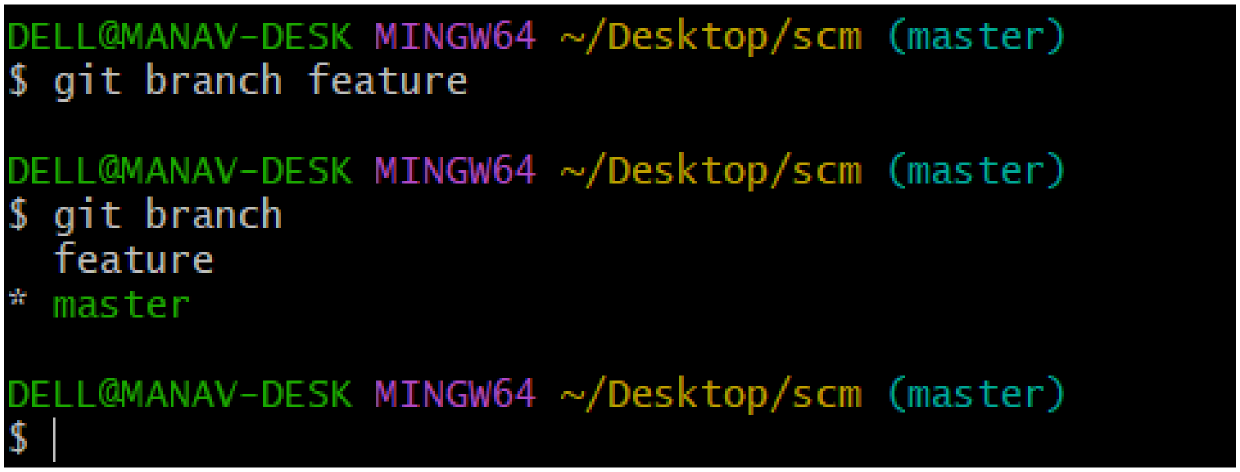
Syntax: For creating a new branch, git branch name by default is master branch.





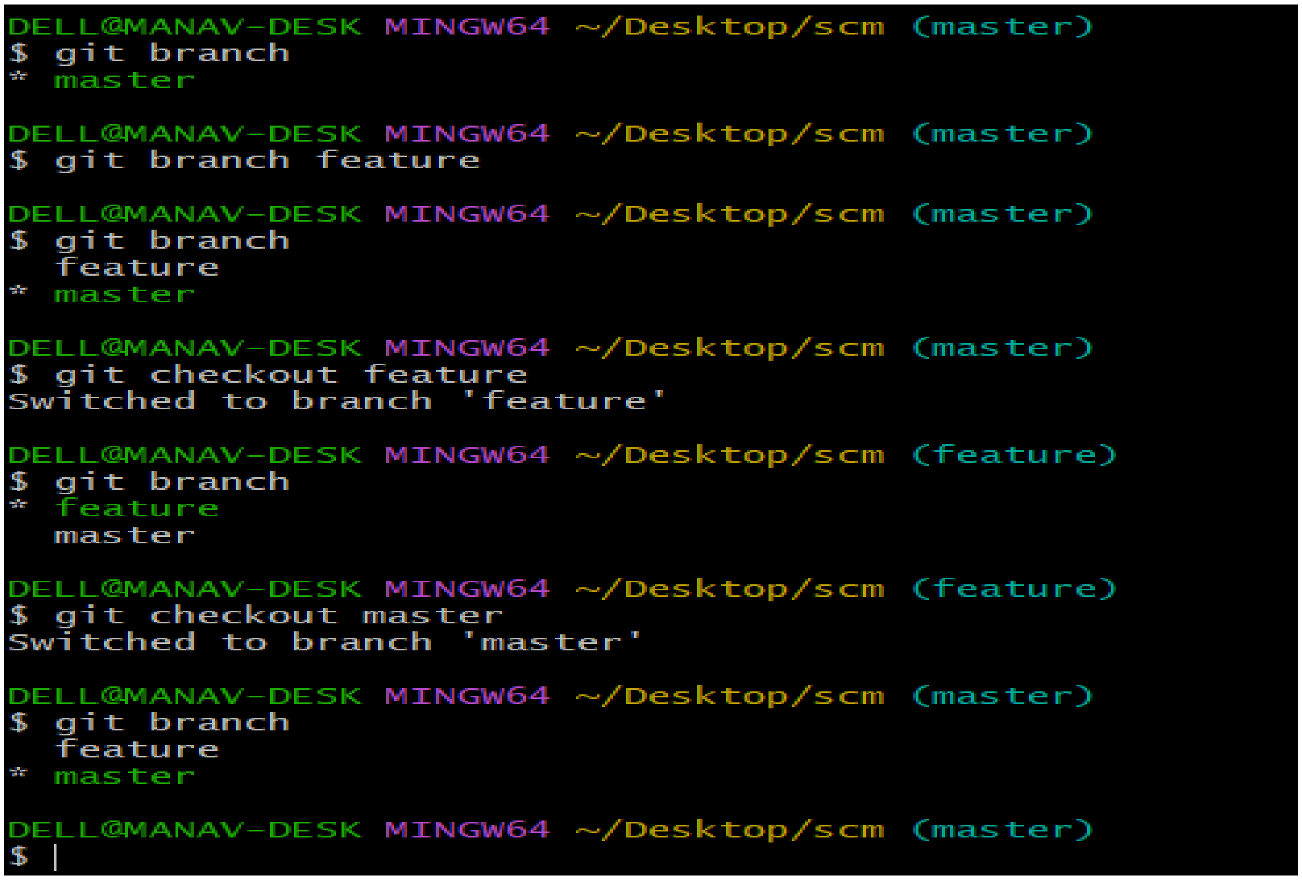
```
DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch
* master

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ |
```

Default branch is master branch



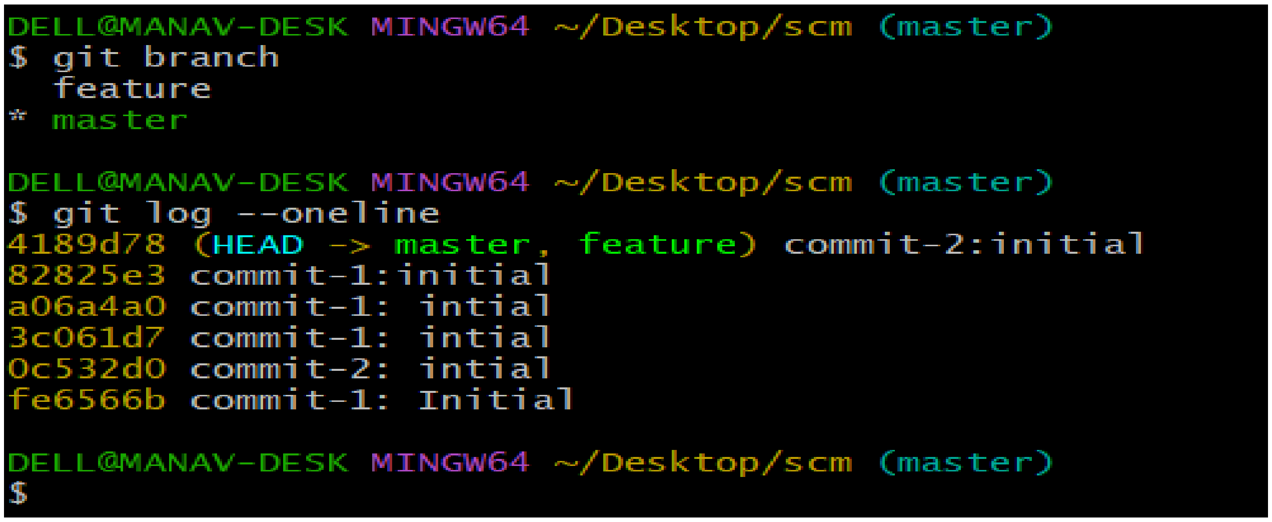
```
DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch feature

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch
  feature
* master

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ |
```

Switching to feature branch



```
DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch
* master

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch feature

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch
  feature
* master

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git checkout feature
Switched to branch 'feature'

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (feature)
$ git branch
* feature
  master

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (feature)
$ git checkout master
Switched to branch 'master'

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch
  feature
* master

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ |
```

Switching to master branch



```
DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git branch
  feature
* master

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$ git log --oneline
4189d78 (HEAD -> master, feature) commit-2:initial
82825e3 commit-1:initial
a06a4a0 commit-1: intial
3c061d7 commit-1: intial
0c532d0 commit-2: intial
fe6566b commit-1: Initial

DELL@MANAV-DESK MINGW64 ~/Desktop/scm (master)
$
```

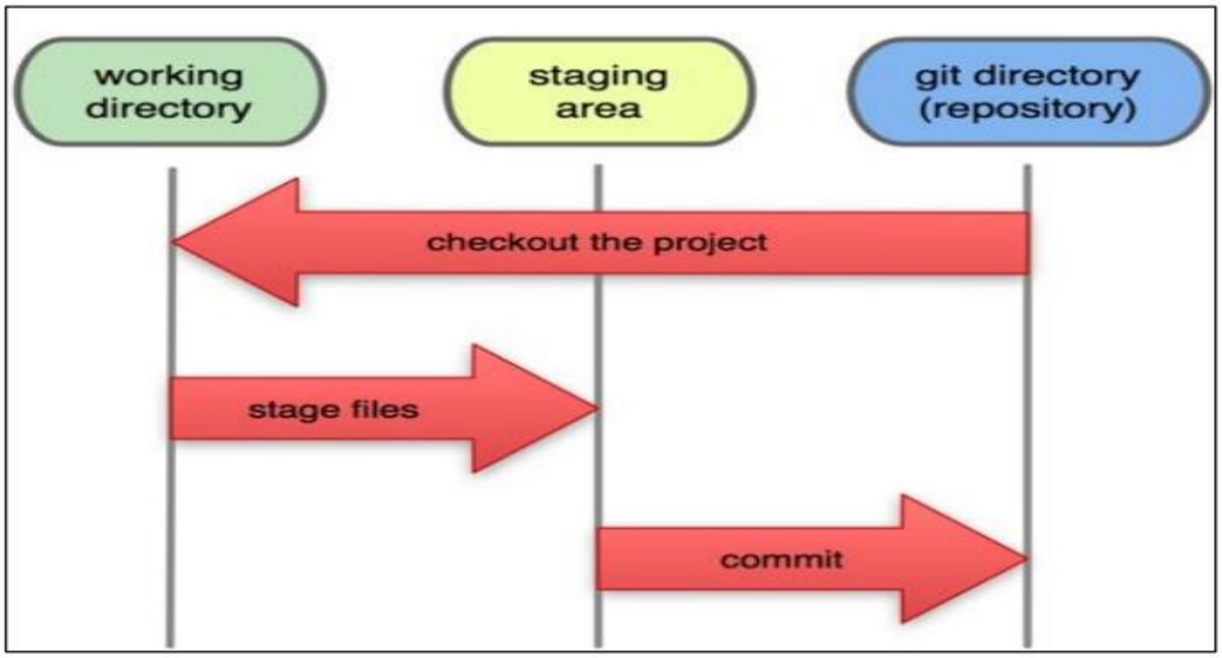
Checking commits

EXPERIMENT-5

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

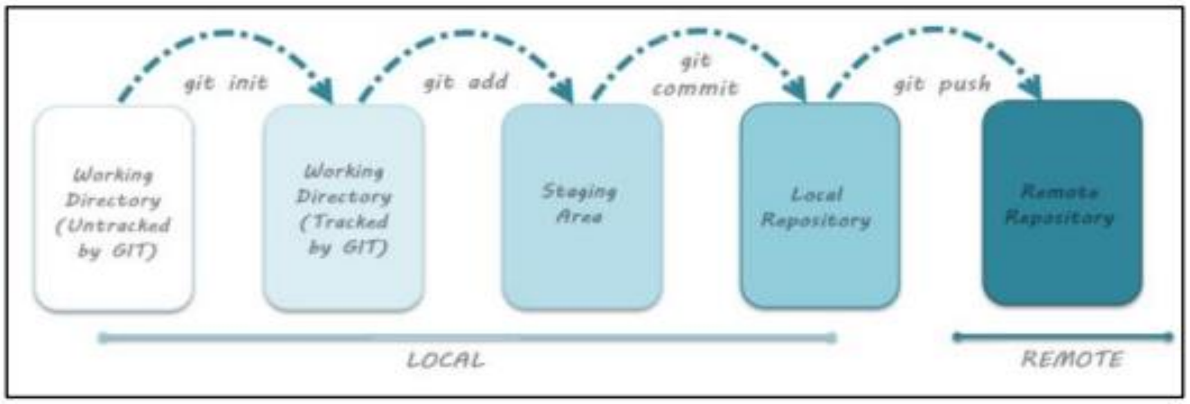


Working Directory: Consider a project residing in your local system. This project may or may not be tracked by Git. In either case, this project directory is called your Working directory.

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.


git init
git add
git commit
git push
Working Directory (Untracked by GIT)
Working Directory (Tracked by GIT)
Staging Area
Local Repository
Remote Repository
LOCAL
REMOTE