Subject Name: **Source Code Management**

Subject Code: **22CS003**

Session: **2022-23**

Department: **DCSE**



**Submitted B: Submitted To:**

Sanisth Tara Shinnu Jangra

2210992247 – First Year

G14 – B

|  |  |
| --- | --- |
| nO | AIM |
| 1 | Setting up of a Git Client. |
| 2 | Setting up of a GitHub Account |
| 3 | How to create, add and commit files and generate logs. |
| 4 | How to create and visualize branches |
| 5 | Explain Git Life Cycle |

[Figure 1 Git link 3](#_Toc126833036)

[Figure 2 Download for windows 4](#_Toc126833037)

[Figure 3 Centralised Version Control System 4](#_Toc126833038)

[Figure 4 Distributed Version Control System 5](#_Toc126833039)

[Figure 5 GitHub Signup Webpage 6](#_Toc126833040)

[Figure 6 GitHub Dashboard 6](#_Toc126833041)

[Figure 7 Creation Of Files 8](#_Toc126833042)

[Figure 8 Usage of Git Add 9](#_Toc126833043)

[Figure 9 Usage of Git Commit 9](#_Toc126833044)

[Figure 10 Usage of Git Log 10](#_Toc126833045)

[Figure 11 Creation of Branch 11](#_Toc126833046)

[Figure 12Changing of Branch 11](#_Toc126833047)

[Figure 13 Git Life Cycle 12](#_Toc126833048)

[Figure 14 Essence of Git 14](#_Toc126833049)

**Experiment 1**

**Aim:** Setting up of a Git Client

**Objective 1:** What is a git?

Git is a version control system used for tracking changes in computer files. It is f=generally used for source code management is software development. Git is used to tracking changes in the source code. The distributed version control tool is used for source code management.

**Objective 2:** Installation of Git.

1. Open ant web browser of your own choice and type <http://git-scm.com> in your address bar which will take you to a webpage like figure 1.1.

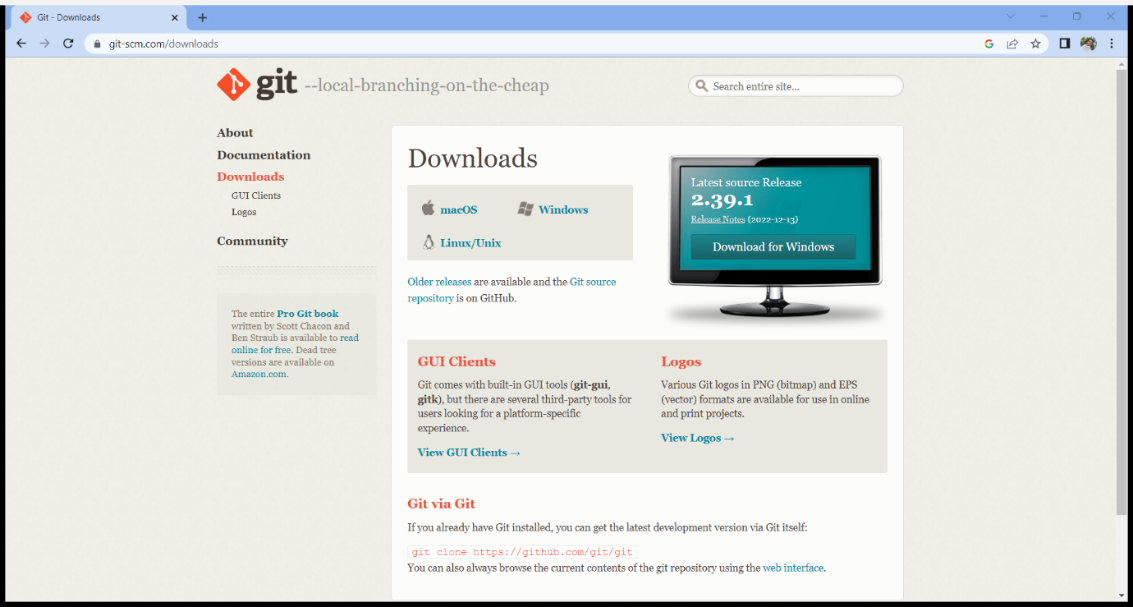


Figure 1 Git link

1. Below Downloads click on the link depending upon the operating system you are currently using.
2. Under the standalone installer section clink on the link depending upon the type of configuration of system you have like 32 bit or 64 bits.

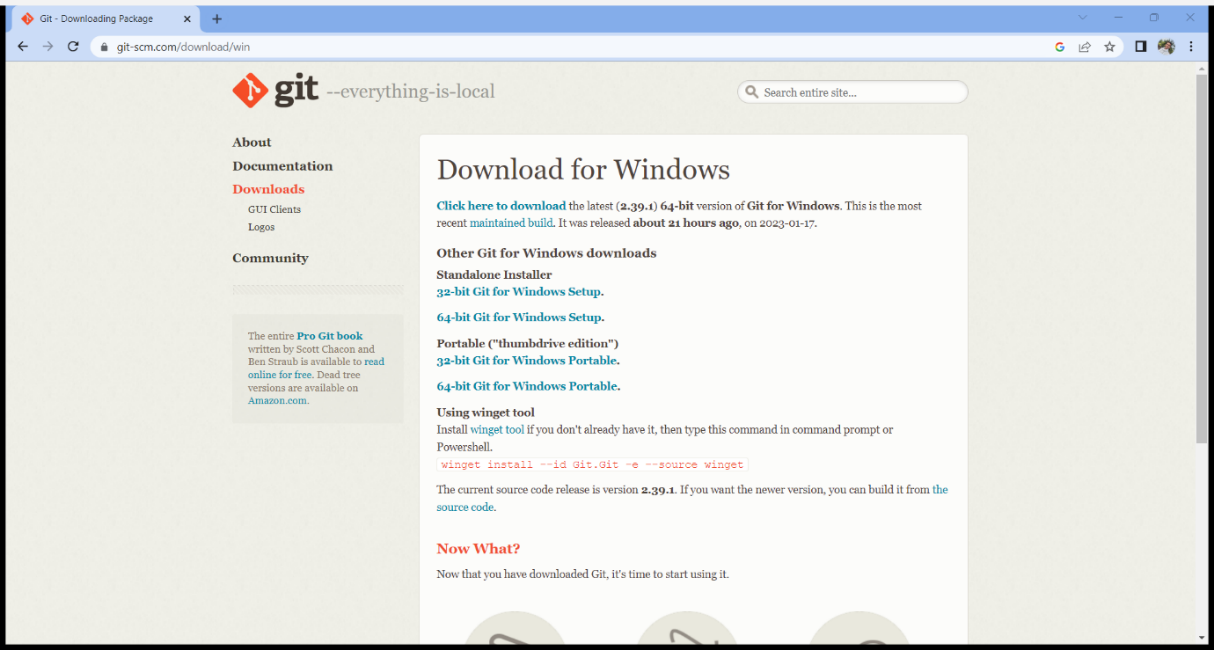


Figure 2 Download for windows

1. Upon clicking on the link, you will install setup from website.
2. Run the setup and install git through recommended settings.

**Objective 3:** What is VCS?

VCS otherwise also called as Version CONTROL system are software tool designed to help teams work in parallel. You can use Version Control for versioning code, binary files, and digital assets. Version Control includes version control software, version control system, or version control tools.

**Objective 4:** What are the types of VCS?

There are two types of VCS:

* Centralised Version Control System (CVCS)

A centralized version control system offers software development teams a way to collaborate using a central server. In a centralized version control system (CVCS), a server acts as the main repository which stores every version of code.

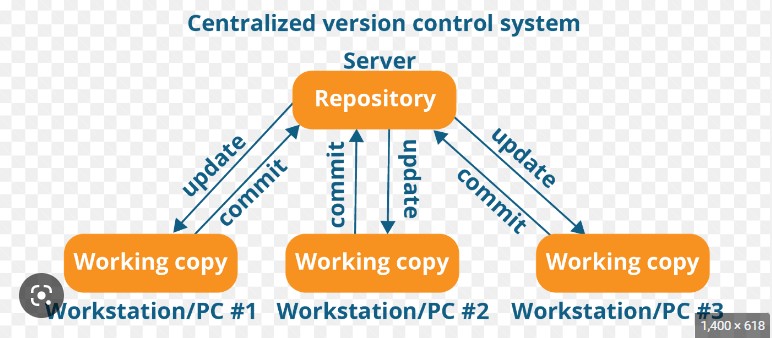


Figure Centralised Version Control System

* Distributed Version Control System (DVCS)

A distributed version control system (DVCS) brings a local copy of the complete repository to every team member's computer, so they can commit, branch, and merge locally.

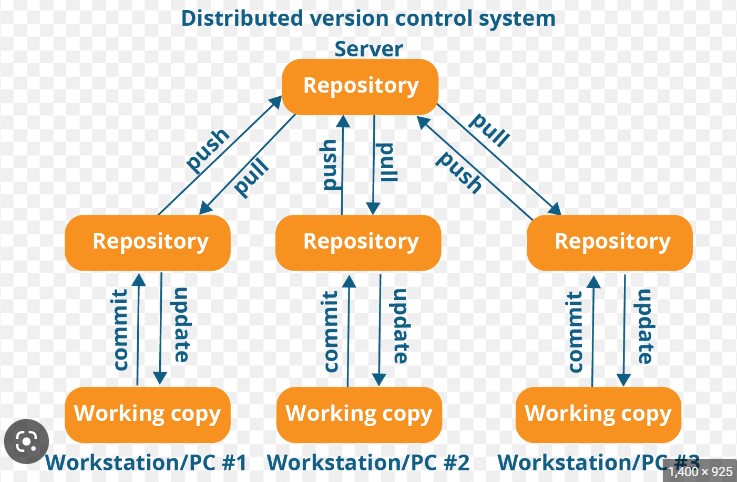


Figure Distributed Version Control System

Git Commands:

* pwd – Present Working Directory
* git -- version – Shows the git version
* ls – Shows the list in the directory or File
* cd – Change Current Working Directory
* cp – Copy
* mv – Move
* mkdir – Make Directory
* rmdir – Remove Directory
* touch – Used to create one or more than one files
* vim “Filename” – Used as text editor
* cat “Filename” – Read the text of file
* git init – Initialize a new repository
* git config – global user.name “Mona Lisa” – Set a name in global scale
* git config – global user.email “2@gmail.com” – Set an email in global scale
* git add “Filename” – file added to the list of changes when we commit
* git commit – m “Comment” – Commit to be done with a comment on commit
* git branch – check the branch
* git branch “Branch Name” – Creates a new branch

**Experiment 2**

**Aim:** Setting up of a GitHub account.

**Objective 1:** Signing up for a GitHub account.

1. Open any web browser of your choice and type <https://github.com/> in the address bar.

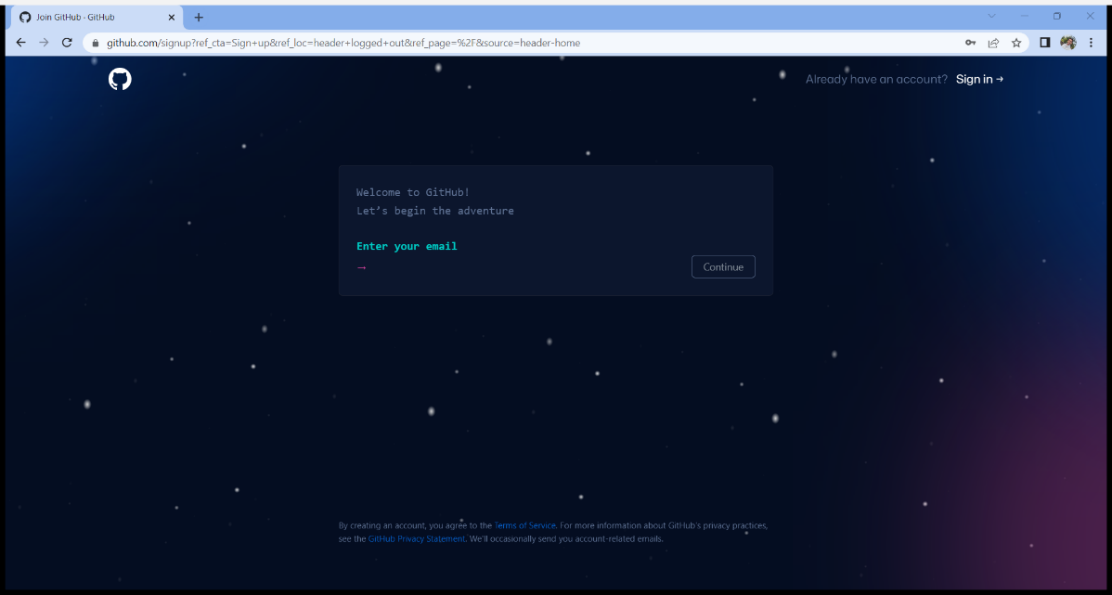


Figure GitHub Signup Webpage

1. Go through the process of signing up an account by filling up the information requested by the website wo sign up.
2. After going through the process and successfully completing it you will find yourself in dashboard.

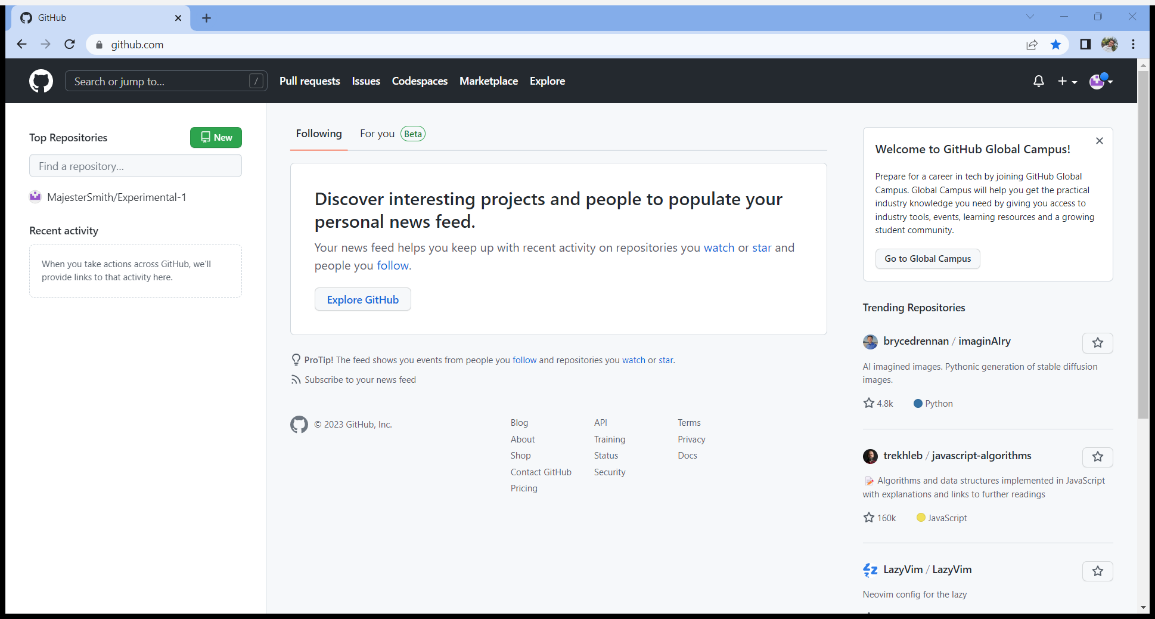


Figure GitHub Dashboard

**Objective 2:** Differentiate between the Git and GitHub.

|  |  |
| --- | --- |
| **Git** | **GitHub** |
| Git is a software. | GitHub is a service. |
| Git is a command-line tool | GitHub is a graphical user interface |
| Git is installed locally on the system | GitHub is hosted on the web |
| Git is maintained by Linux. | GitHub is maintained by Microsoft. |
| Git is focused on version control and code sharing. | GitHub is focused on centralized source code hosting. |
| Git is a version control system to manage source code history. | GitHub is a hosting service for Git repositories. |

**Objective 3:** Explain repository and their types.

A repository contains all of your project's files and each file's revision history. You can discuss and manage your project's work within the repository.

There are two types of Repositories –

* Bare Repositories
* Non – Bare Repositories

Bare Repositories: These repositories are used to share the changes that are done by different developers. A user is not allowed to modify this repository or create a new version for this repository based on the modifications done.

Non-bare Repositories: Non-bare repositories are user-friendly and hence allow the user to create new modifications of files and also create new versions for the repositories. The cloning process by default creates a non-bare repository if any parameter is not specified during the clone operation.

**Experiment 3**

**Aim:** How to create, add and commit files and generate logs.

**Objective 1:** Creation of Files.

We can use command ‘touch “Filename.txt”’ to create a file in the working directory of your own choice.

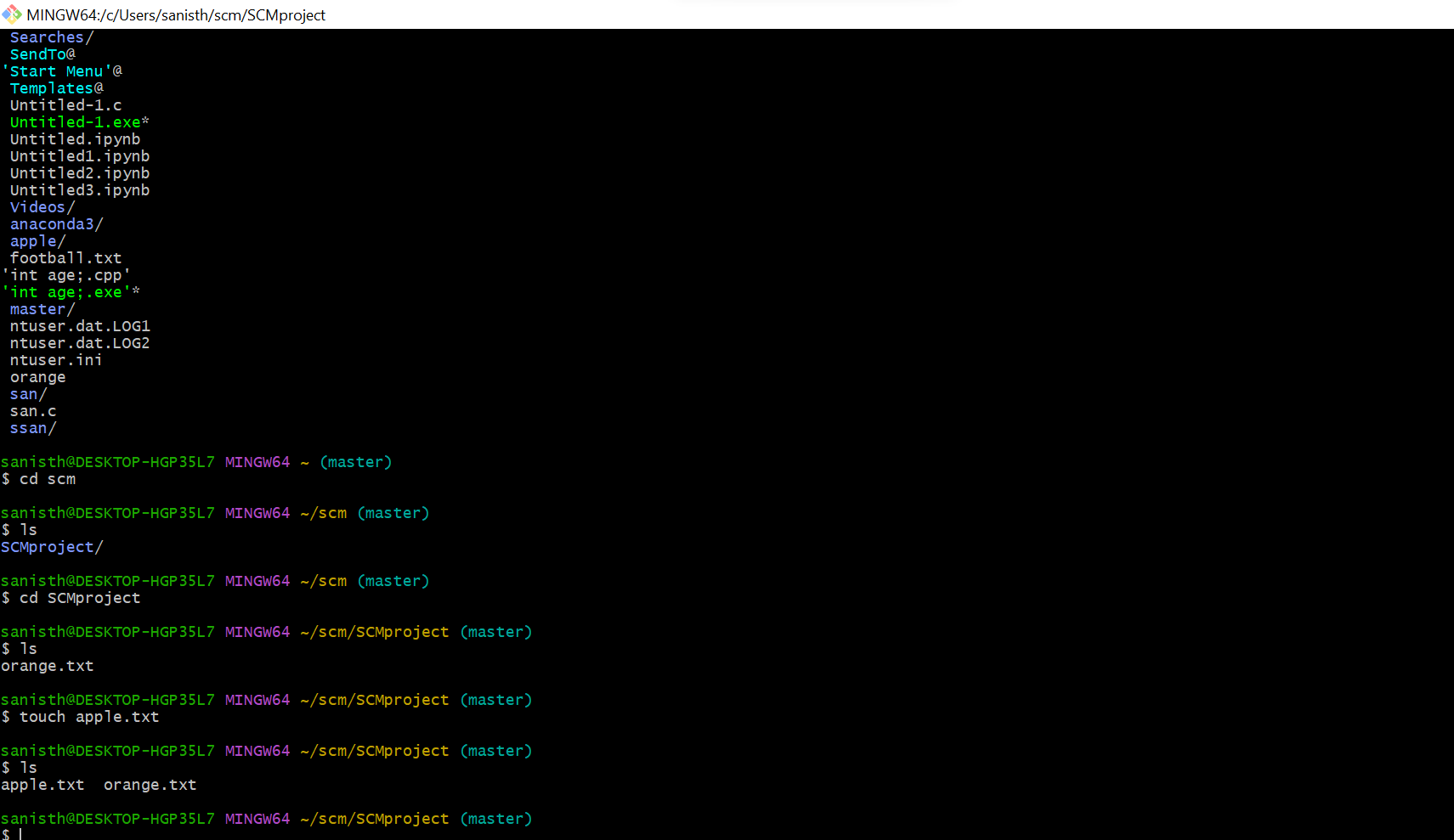
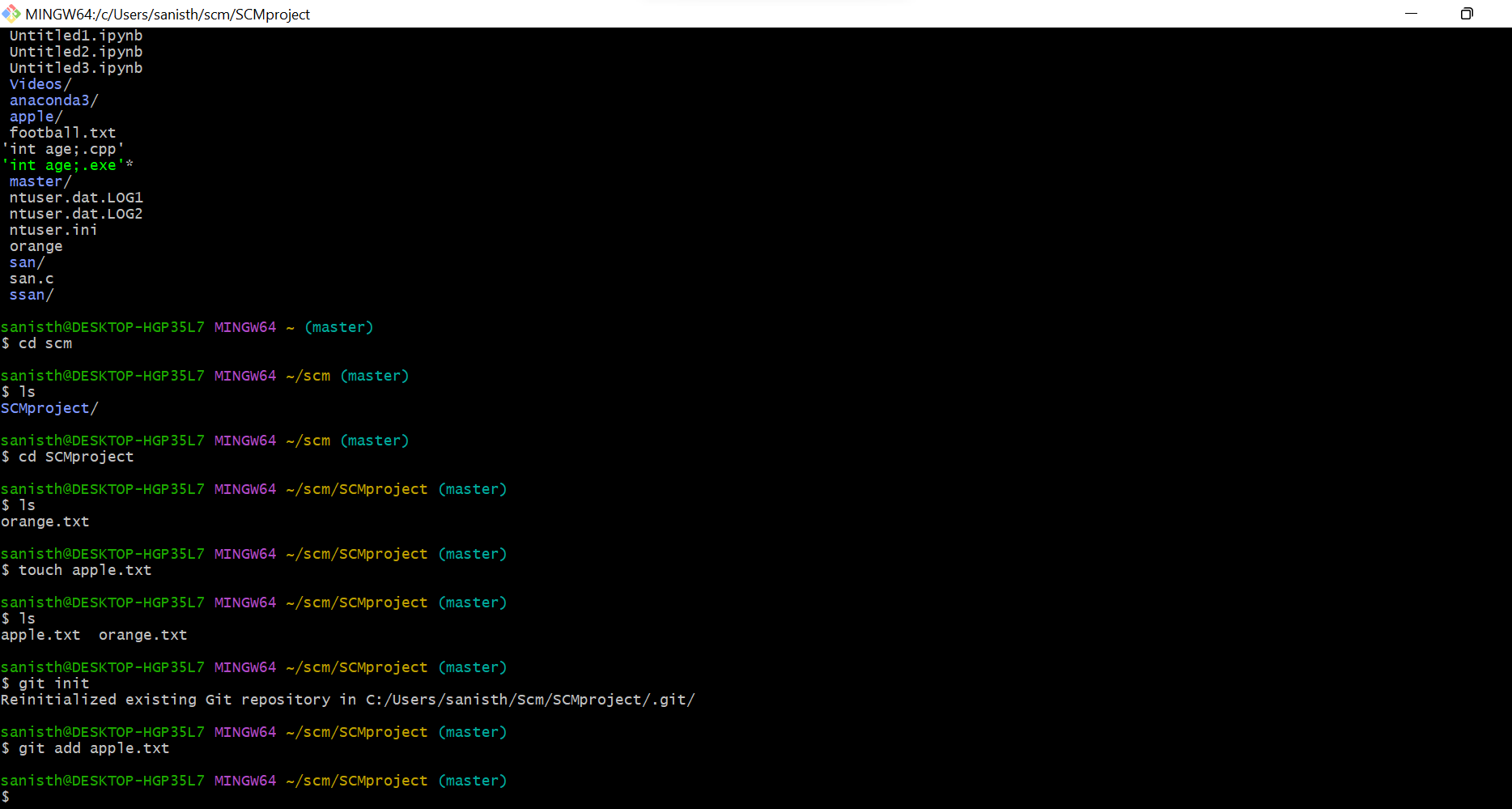


Figure Creation Of Files

**Objective 2:** Adding of files to commit.

We can use command ‘git add “Filename.txt”’ to add the file to list of changes to be made when we commit.

Be sure to use ‘git init’ command before using ‘git add’ command to avoid unnecessary troubles which might come up later.

Figure Usage of Git Add

**Objective 3:** Committing the files and checking the history of commit through logs.

We can commit the Files by using ‘git commit -m “Comment”’ command which allows us to commit the files which were added using ‘git add’ command before.

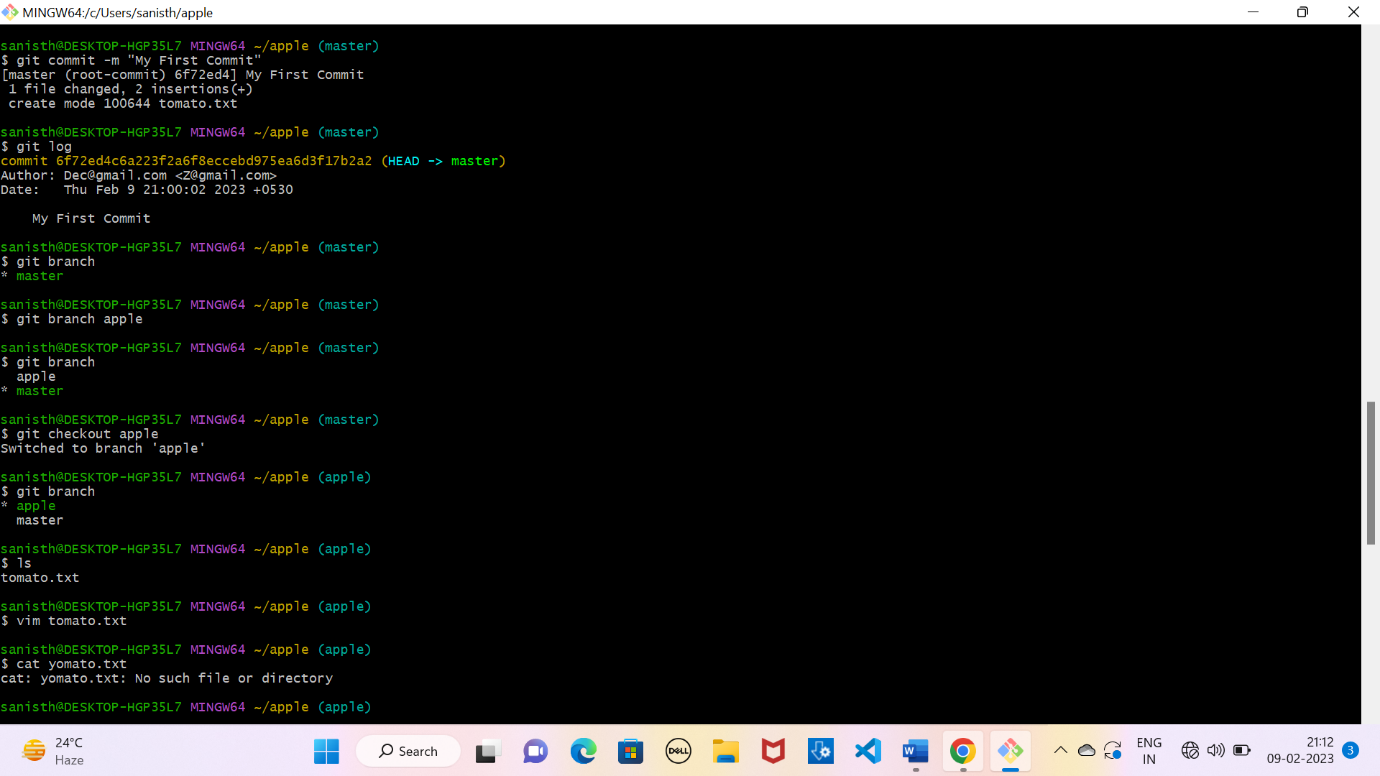


Figure Usage of Git Commit

We can see the history of commits made by using ‘git log’ command which allows us to see the commits we made.

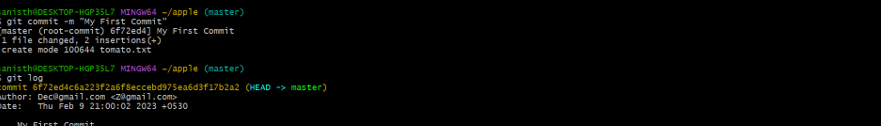


Figure Usage of Git Log

**EXPERIMENT-4**

**Aim:** How to create and visualise branches.

**Objective 1:** Creation of branches in git.

We can use ‘git branch’ command to see the number of branches of git whereas we can use ‘git branch “Branch name”’ to create a new branch in git.’

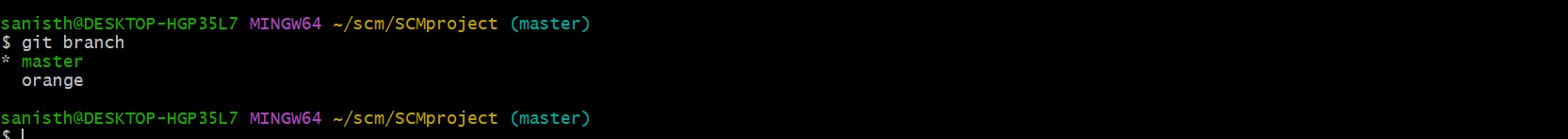


Figure Creation of Branch

**Objective 2:** Changing of branch.

We can change from one branch to another with the help of ‘git checkout “Branch name”’ command which allows use to navigate between bit branches.

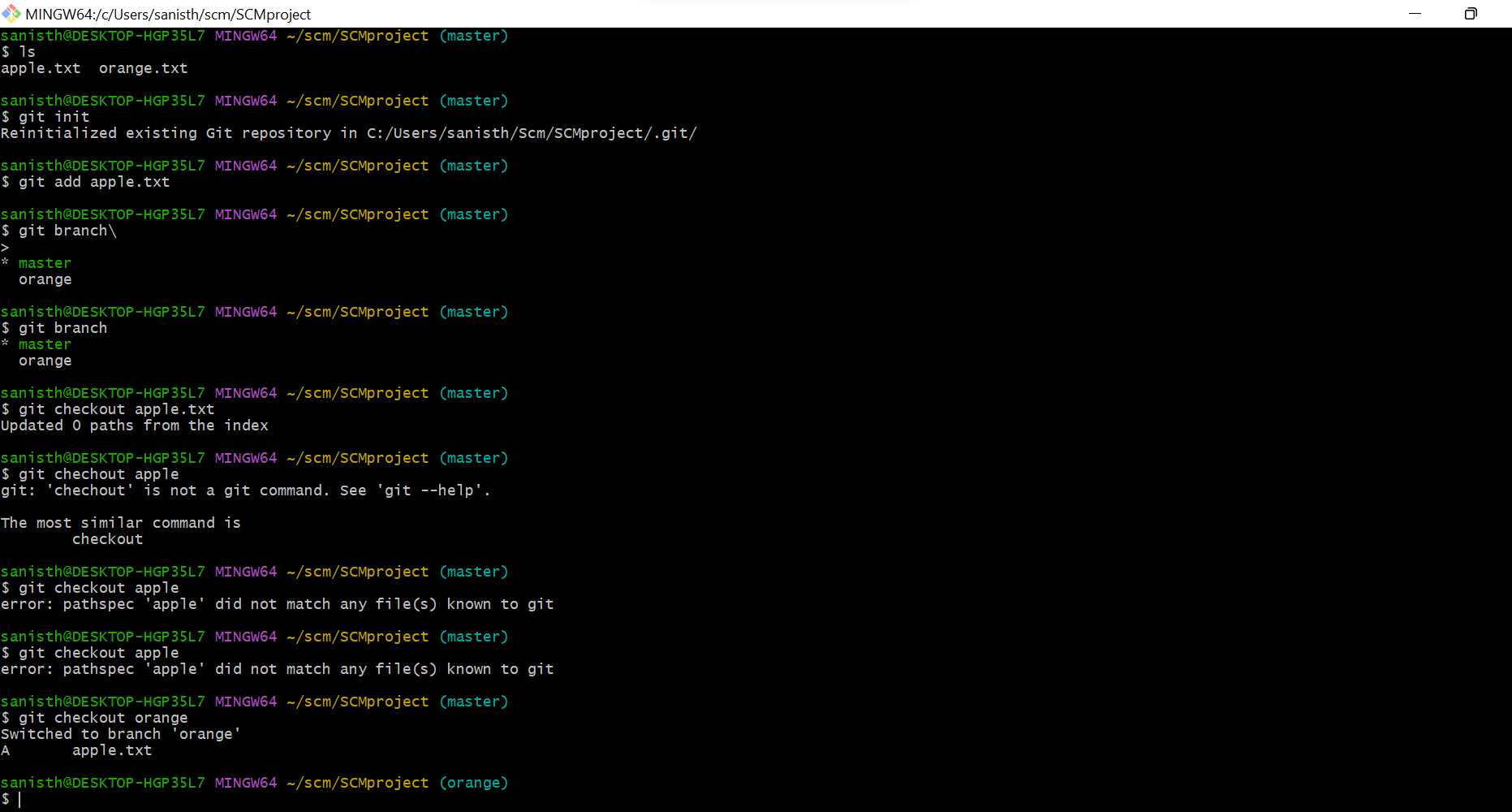


Figure Changing of Branch

**Experiment 5**

**Aim:** Explain Git life cycle.

**Objective 1:** Explanation of Git life Cycle.

We can define workflow of Git Life cycle as follows:

* We will create a branch on which we can work on and later we will merge it with master
* Clone: First, when we have code present in the remote repository, we clone to local to form something called a local repository.
* Modifications/Adding Files: we perform several developments on the existing files or may as well add new files. Git will monitor all these activities and will log them.
* We need to move the content that we require to transform to the master to the staging area by using git commands and the snapshot of staged files will be saved in the git staging area.
* We need to perform commits on the files that are staged and the recorded snapshot from the above steps will be permanently saved on the local repo and this particular is recorded by commit message for future referrals.
* Once we commit the code is available on the local repo but to send it to the master repo, we need to perform PUSH operation
* If someone else is working on the same branch then there will be a possibility that he might have added his changes to the master by push. So, we need to perform PULL operation before the PUSH operation if multiple people are working on the same branch.
* Once the target branch is updated, we need to get all the required approvals so that merge operation with the master is allowed.

This is the basic workflow of git, and there are lots of intermediate commands like git add, git status, git commit, git push origin, git rebase, git merge, git diff, etc which can be used depending upon the requirement of the user.

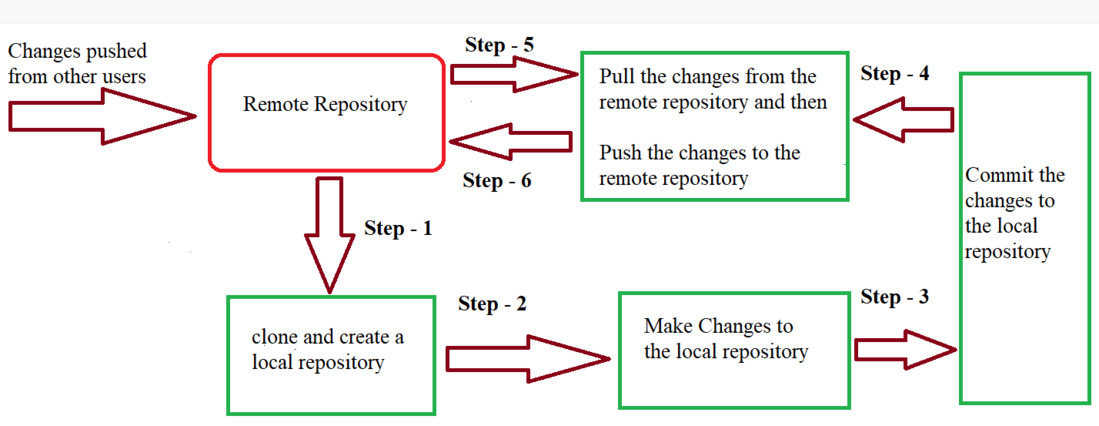


Figure Git Life Cycle

**Objective 2:** Define states which makes the essence of Git Version Control System.

There are three states which makes the essence of Git Version Control system.

* Working Directory
* Staging Area
* Git Directory

**Working Directory:**

Whenever we want to initialize our local project directory to make it a git repository, we use the git init command. After this command, git becomes aware of the files in the project although it doesn’t track the files yet. The files are further tracked in the staging area.

**Staging Area:**

Now, to track the different versions of our files we use the command git add. We can term a staging area as a place where different versions of our files are stored. git add command copies the version of your file from your working directory to the staging area. We can, however, choose which files we need to add to the staging area because in our working directory there are some files that we don’t want to get tracked, examples include node modules, env files, temporary files, etc. Indexing in Git is the one that helps Git in understanding which files need to be added or sent. You can find your staging area in the. git folder inside the index file.

**Git Directory:**

Now since we have all the files that are to be tracked and are ready in the staging area, we are ready to commit our files using the git commit command. Commit helps us in keeping the track of the metadata of the files in our staging area. We specify every commit with a message which tells what the commit is about. Git preserves the information or the metadata of the files that were committed in a Git Directory which helps Git in tracking files and basically it preserves the photocopy of the committed files. Commit also stores the name of the author who did the commit, files that are committed, and the date at which they are committed along with the commit message.

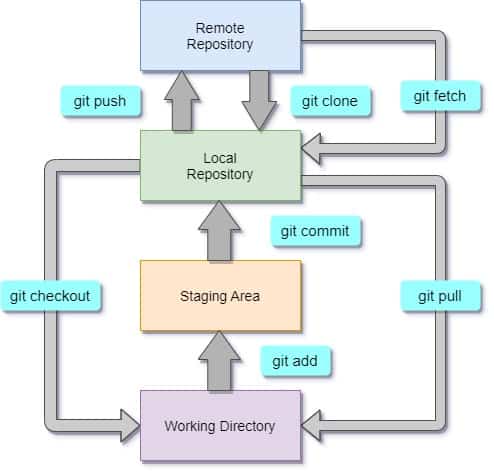


Figure Essence of Git