Subject Name: **Source Code Management**

Subject Code: **22CS003**

Session: **Jan - June 2024**

Department: **DCSE**

Submitted By:

Yash Kumar

2310992325-First Year

G26-B

Submitted To:

Ms. Gunjan Thakur

List of Programs

Experiment 1

Aim: Setting Up of Git Client

Theory:

Git is a distributed version control system used for tracking changes in source code during software development. It was created by Linus Torvalds in 2005 to manage the development of the Linux kernel but has since become widely adopted in various industries and projects. Git allows multiple developers to collaborate on a project by providing a history of changes made to the codebase. It also provides a powerful set of tools for collaboration, version control, and project management, making it an essential tool for software development teams of all sizes. We can use git and GitHub parallelly to work with multiple people or individually.

Procedure:

We can install Git for Windows through many ways, but the most popular and easy install method is by simply downloading it from the Git website. The steps to download git via Git-website, are as follows:

- Go to https://git-scm.com/downloads or search git-scm on any search engine.
- Select a platform, on which git is to be downloaded, like macOS, Windows or Linux. Since we are, setting git on Windows, we'll click on the Windows option.
- Select the desired bit version and click on the "Click here to download" option.
- The git.exe file will start downloading on your system.

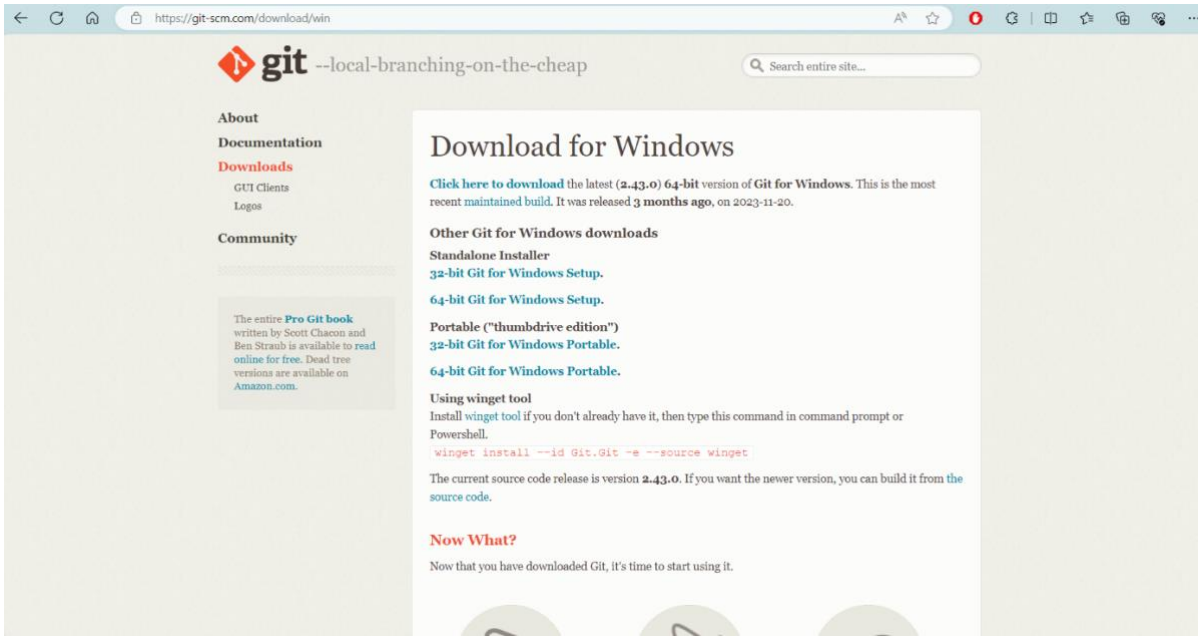

Fig 1.1 Snapshot of download page

Once the git.exe file is installed on your system, open the file, a git install dialog box will appear, as shown in fig 1.2.

Fig 1.2 Git Dialog Box

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.

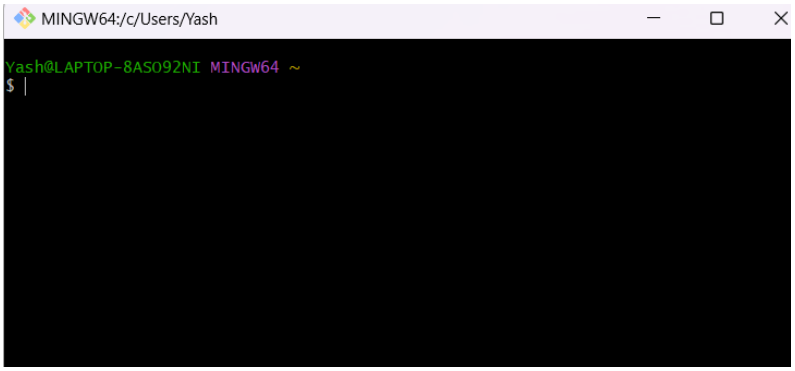

Fig 1.3 Git Bash Window

Next, setup your account name and email by writing the commands, shown in fig 1.4, in your git bash window.

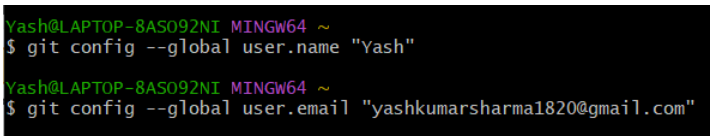

Fig 1.4

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

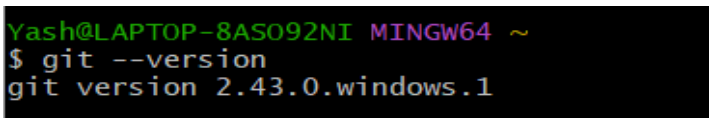

Fig 1.5

Experiment 2

Aim: Setting Up GitHub Account

Theory:

GitHub is a web-based platform for hosting Git repositories and facilitating collaboration among developers. It provides a range of features to help developers and teams manage their code projects effectively. GitHub allows users to host Git repositories on its platform. Developers can create repositories to store their code, documentation, and other project-related files. GitHub offers various collaboration tools to help teams work together on projects. These include pull requests, code reviews, issue tracking, and project boards. GitHub has a large and active community of developers who contribute to open-source projects hosted on the platform. Users can follow repositories, star projects, and fork repositories to create their own copies. GitHub also supports social features like user profiles, followers, and activity feeds.

Procedure:

Creating a GitHub account is a straightforward process, the steps do so are as follows,

- Go to https://github.com or search "GitHub" on any search engine. The following window will appear,

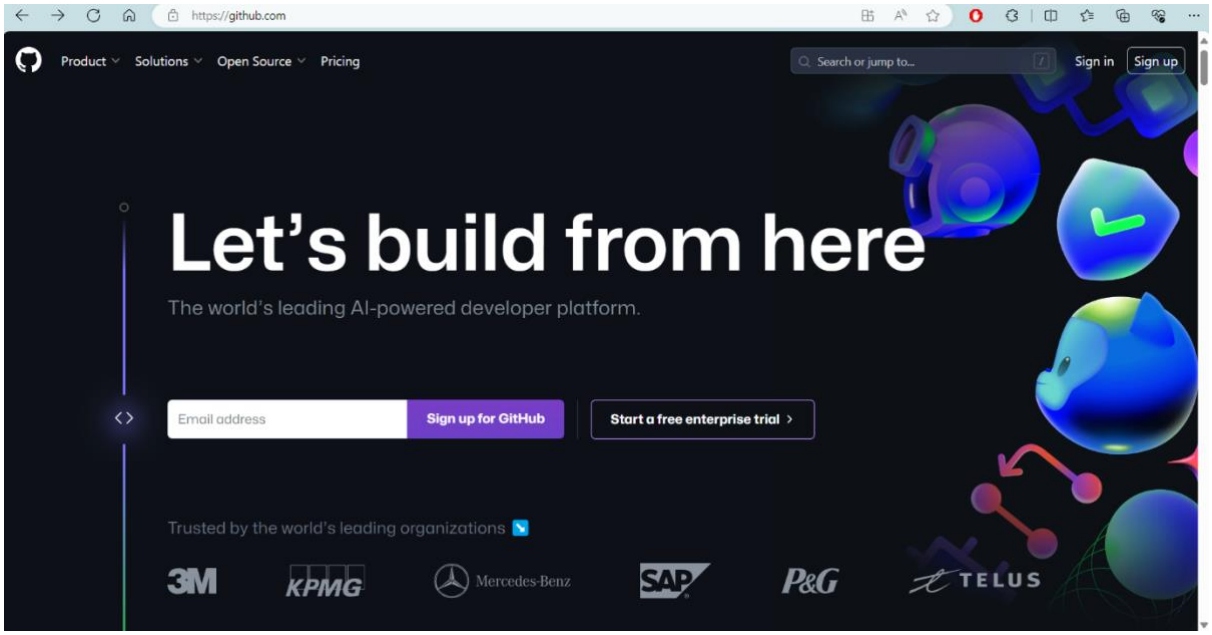

Fig 2.1 GitHub home page

- Enter your email address in the signup field and click on "Sign up for GitHub" button.
- A signup dialog box will appear, follow the signup instructions and complete all the processes.
- This will create your GitHub account and the following window will appear,

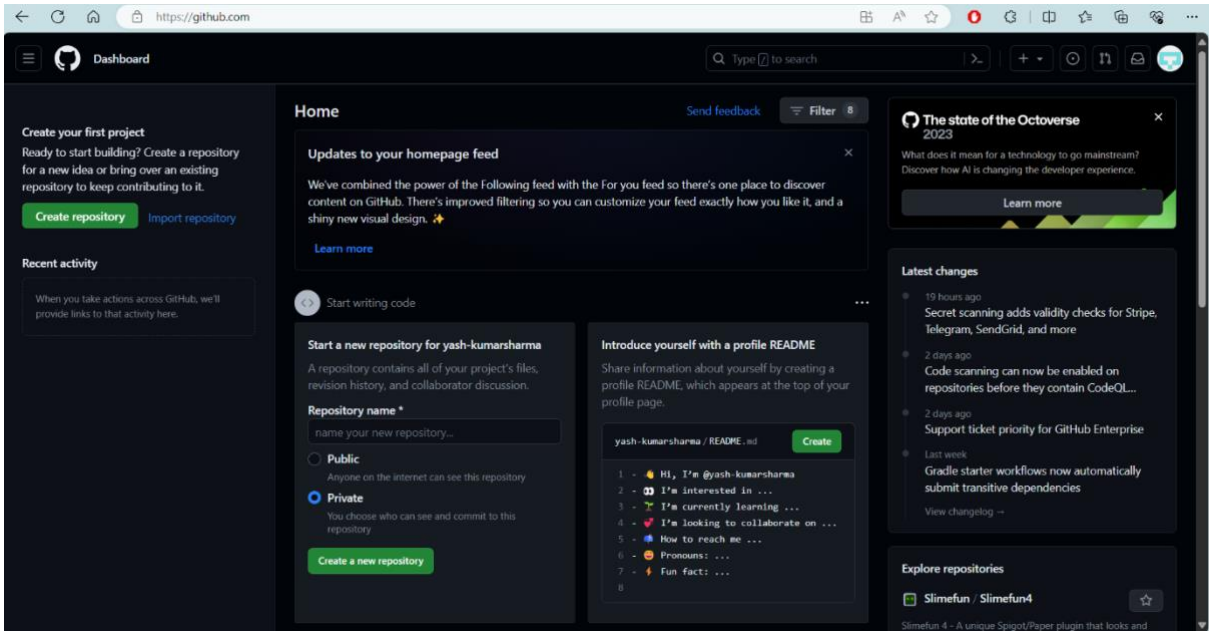

Fig 2.2 GitHub interface

Hence, this way you can easily setup your GitHub account.

Experiment 3

Aim: Generating Logs

Theory:

Logs refer to the recorded history of commits made to a repository. The log provides a chronological record of all the commits, along with relevant information such as the commit message, author, date, and unique commit hash. The commit log is a valuable tool for understanding the history of changes in a Git repository. It allows developers to track modifications, review previous work, and collaborate effectively with others. It displays the commit history in a text-based graphical representation, showing the branching and merging of commits. It provides a visual overview of the repository's history, including branches and their relationships.

Procedure:

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

- Create a new folder in your local drive, and right click it, click on “Open Git Bash here”. This will open the git bash terminal.

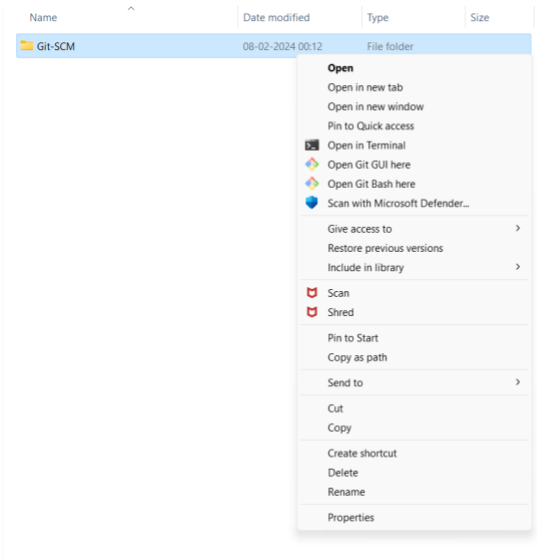

Fig 3.1

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

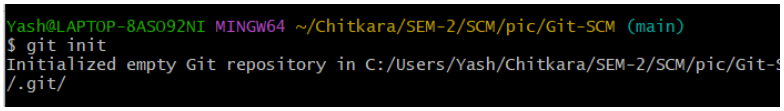

Fig 3.2 git init command interface

- 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>", 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.

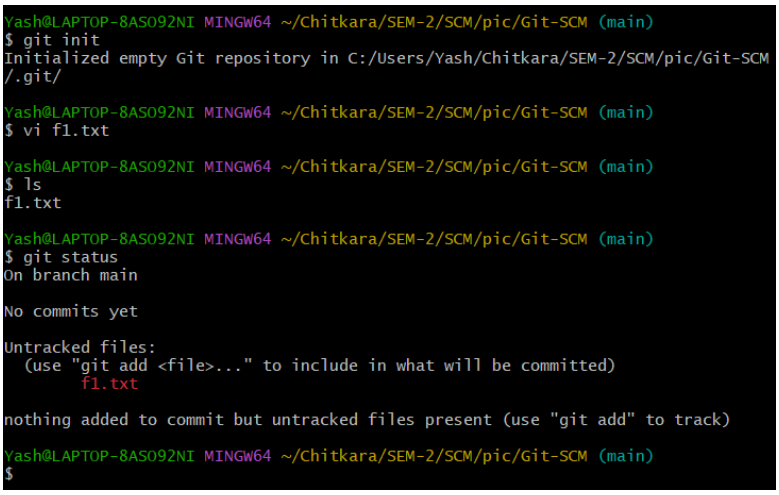

Fig 3.3 Git Command(s) interface - I

- Now we'll 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, "git commit -m "message for commit".
- Now, we'll finally use the "git log" command to display the commit history of the repository.

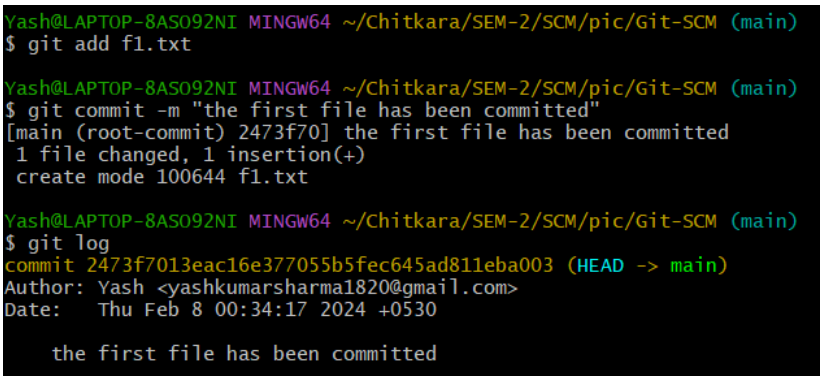

Fig 3.4 Git Command(s) interface - II

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.

Experiment 4

Aim: Create and Visualize Branches

Theory:

In Git, branches are independent lines of development within a repository. Each branch represents a separate sequence of commits that diverge from the main line of development, often called the "master" branch or, more recently, the "main" branch. Branches are a fundamental concept in Git that enable flexible, collaborative, and organized development workflows. They allow developers to work on different tasks concurrently, experiment with changes without risk, and manage complex projects with ease.

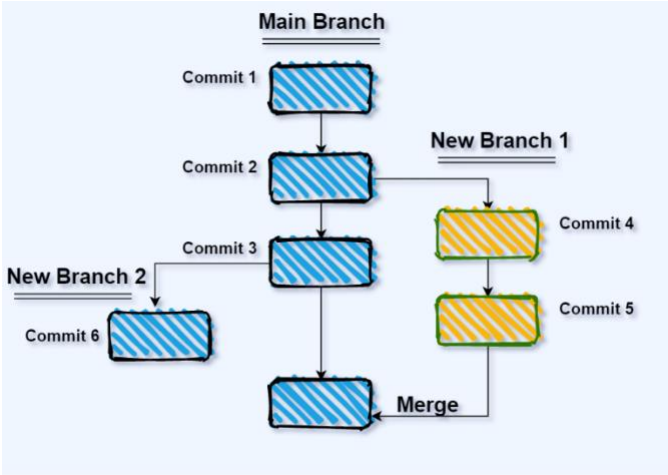

Fig 4.1 Git Branching

The main branch is the default branch in a Git repository, typically named "master" or "main." It represents the primary line of development and serves as the baseline from which other branches diverge.

Procedure:

Newer branches, in git, can be created by using the command, “git branch <branch_name>”. We can create as many as we need, branches. To switch between the created branches we use the command, “git checkout <branch_name>” or “git switch <branch_name>”, this will change our branch from master/main to the one we want to work on.

```
Yash@LAPTOP-8ASO92NI MINGW64 ~/Chitkara/SEM-2/SCM/pic/Git-SCM (main)
$ git branch b1

Yash@LAPTOP-8ASO92NI MINGW64 ~/Chitkara/SEM-2/SCM/pic/Git-SCM (main)
$ git checkout b1
Switched to branch 'b1'

Yash@LAPTOP-8ASO92NI MINGW64 ~/Chitkara/SEM-2/SCM/pic/Git-SCM (b1)
$ git switch main
Switched to branch 'main'
```

Fig 4.2 Git Branching Command(s) - I

To rename a branch we use the command, “git branch -m <old_name> <new_name> and to list all the branch names that we have created or are existing,

we use the command, “git branch” or “git branch –list”. The “*” represents the current branch in which we are working.

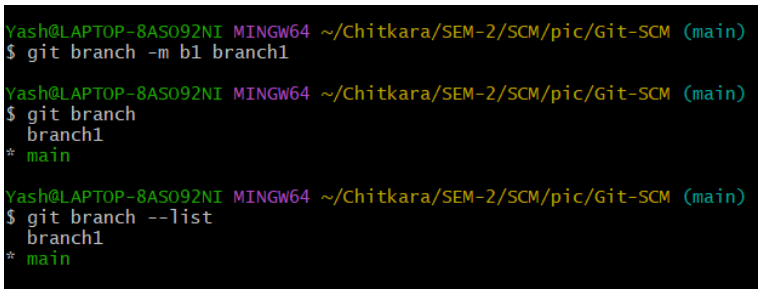

Fig 4.3 Git Branching Command(s) – II

To merge a branch into another, we use the command “git merge <branch_name>”, by staying the main branch. We can also delete a branch, after the completion of our work in it by using the command, “git branch -d <branch_name>”. And, to verify this we can again list all the branches.

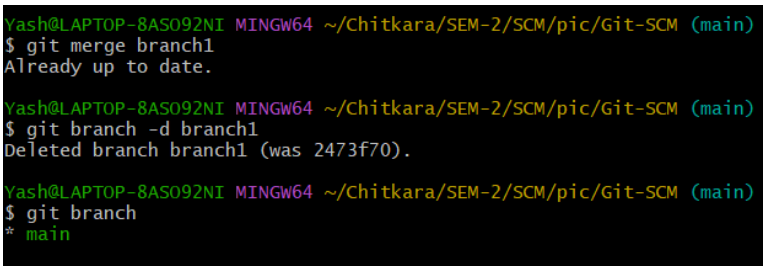

Fig 4.4 Git Branching command(s)

Hence, this way we created and visualized branches in git and used various git branching commands.

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.

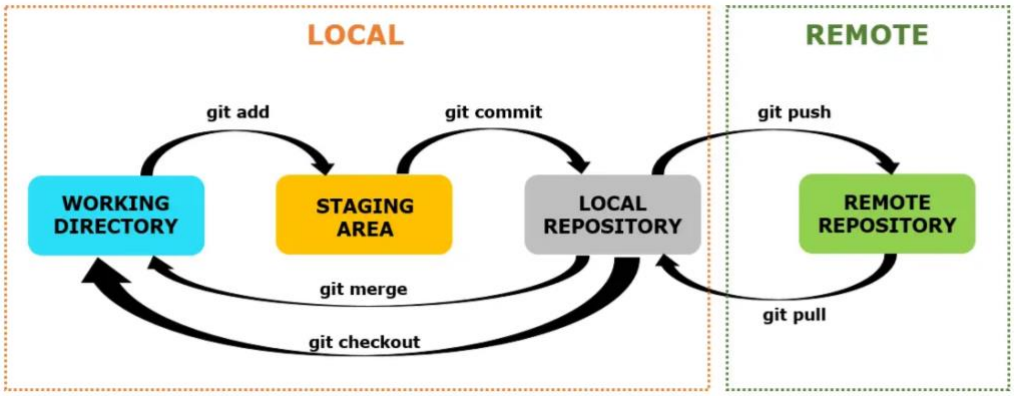

Fig 5.1 Git Lifecycle

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.

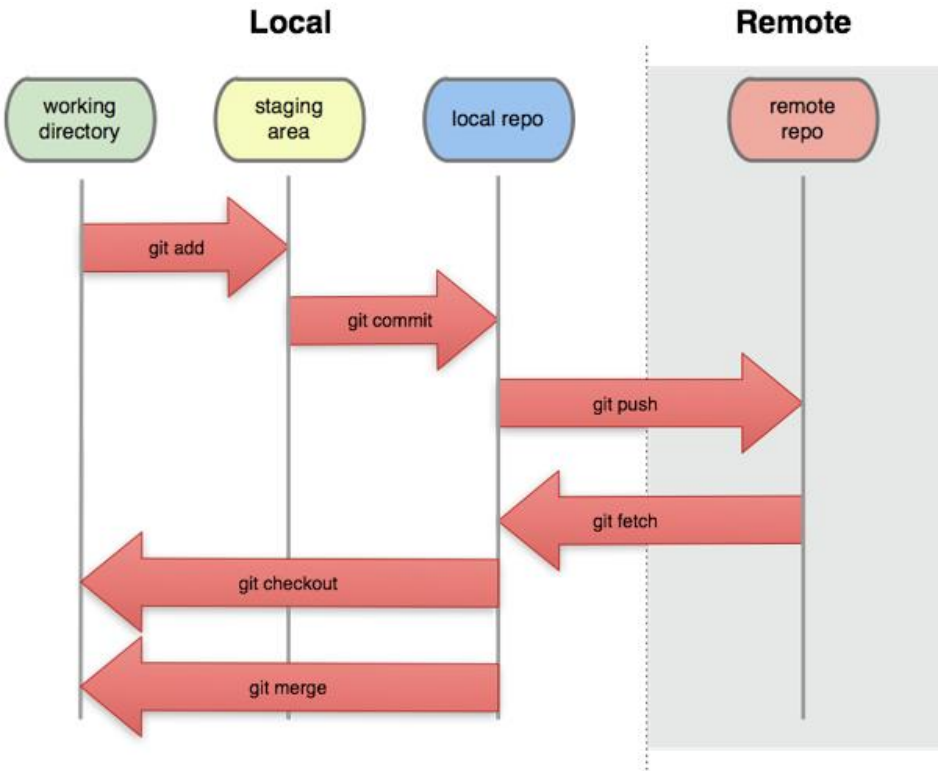

Fig 5.2 Major Git Sections