Name - Parth Mahesh Madrewar
Reg, No. - 241080036
Experiment No. - 3

Aim : To learn and apply fundamental Git commands such as init, add, commit, branch, and merge for monitoring and handling code modifications.

What is Git ?

• Introduction
Git is a Distributed Version Control System (DVCS) that helps developers track and manage changes in project files, especially code. In DVCS, every developer has the full history of the project on their own computer — not just on one central server.

• GitHub
GitHub is an online platform where you can store Git repositories in the cloud. It makes teamwork easier by adding features like:

- Project planning
- Code reviews
- Issue tracking
- Team collaboration
- Open source contributions

Basic Git/GitHub Commands :

1. git init
Initializes a new Git repository in the current directory by creating a hidden .git folder. This folder stores all the version control data.
2. git status
Shows the current state of the working directory and staging area. It helps identify which files are staged, modified, or untracked.
3. touch filename
Creates a new empty file named filename. Commonly used to simulate file creation for testing or development.
4. git add <filename>
Stages the specified file for the next commit. Git will track changes in that file when committing.
5. git commit -m "message"
Commits the staged changes to the local repository with a custom message describing the changes.
6. git config --global user.email "you@example.com"
Sets the email address for Git to use in all commits made from the system. This is part of the user identity.
7. git config --global user.name "Your Name"
Sets the Git username globally for all repositories. Both name and email appear in the commit metadata.
8. git config user.name
Displays the current configured username in the local or global Git settings.
9. git config user.email
Displays the configured email address used by Git for commits.
10. git add .
Stages all changed and new files in the current directory and subdirectories for the next commit.

11. git branch
Lists all existing branches in the repository and highlights the current one.
12. git branch <branchname>
Creates a new branch with the given name. Branches allow working on separate features or versions without affecting the main codebase.
13. git checkout <branchname>
Switches the working directory to the specified branch. Future commits will be made to this branch.
14. git log
Displays a history of commits, including author info, date, commit hash, and message.
15. git add <another-file>
Stages an additional file for the next commit, useful when adding new files after initial staging.
16. git show
Shows detailed information about a specific commit, including file changes and diff outputs.
17. git checkout master
Switches back to the master branch from another branch.
18. git merge <branchname>
Merges changes from the specified branch into the current branch. This integrates features or updates developed in separate branches.

Screenshots of every command:

```
sysadmin@sysadmin:~/parth/parth_OST$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.
```

```
sysadmin@sysadmin:~$ git config --global user.name"parth13m"
sysadmin@sysadmin:~$ git config --global user.email"parthmadrewar13@gmail.com"
sysadmin@sysadmin:~$
```

```
sysadmin@sysadmin:~/git-lab$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/sysadmin/git-lab/.git/
sysadmin@sysadmin:~/git-lab$ git add file1.txt
sysadmin@sysadmin:~/git-lab$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt

sysadmin@sysadmin:~/git-lab$
```

```
sysadmin@sysadmin:~/git-lab$ echo "this is second file" > file2.txt
sysadmin@sysadmin:~/git-lab$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add
sysadmin@sysadmin:~/git-lab$ git add
Nothing specified, nothing added.
hint: Maybe you wanted to say 'git add .'?
hint: Turn this message off by running
hint: "git config advice.addEmptyPathspec false"
sysadmin@sysadmin:~/git-lab$ git add file2.txt
sysadmin@sysadmin:~/git-lab$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   file1.txt
        new file:   file2.txt

sysadmin@sysadmin:~/git-lab$
```

```
sysadmin@sysadmin:~/git-lab$ git config --global user.name "parth13m"
sysadmin@sysadmin:~/git-lab$ git config --global user.email "parthmadrewar13@gmail.com"
sysadmin@sysadmin:~/git-lab$
```

```
sysadmin@sysadmin:~/git-lab$ git commit -m "Initial commit with file1 and file2"
[master (root-commit) 0207864] Initial commit with file1 and file2
 2 files changed, 2 insertions(+)
 create mode 100644 file1.txt
 create mode 100644 file2.txt
```

```
sysadmin@sysadmin:~/git-lab$ git branch b1
sysadmin@sysadmin:~/git-lab$ git checkout b1
Switched to branch 'b1'
sysadmin@sysadmin:~/git-lab$ git status
On branch b1
nothing to commit, working tree clean
sysadmin@sysadmin:~/git-lab$
```

```
nothing to commit, working tree clean
sysadmin@sysadmin:~/git-lab$ echo "adding in b1" >> file1.txt
sysadmin@sysadmin:~/git-lab$ git add .
sysadmin@sysadmin:~/git-lab$ git commit -m "changes in b1"
[b1 2c9a220] changes in b1
 1 file changed, 1 insertion(+)
sysadmin@sysadmin:~/git-lab$
```

```
sysadmin@sysadmin:~/git-lab$ git checkout master
Switched to branch 'master'
sysadmin@sysadmin:~/git-lab$ git merge b1
Updating 0207864..2c9a220
Fast-forward
 file1.txt | 1 +
 1 file changed, 1 insertion(+)
sysadmin@sysadmin:~/git-lab$
```

```
sysadmin@sysadmin:~/git-lab$ git log
commit 2c9a220f668488b78e87590389b13d23a767e1d3 (HEAD -> master, b1)
Author: parth13m <parthmadrewar13@gmail.com>
Date:   Wed Aug 6 16:44:21 2025 +0530

    changes in b1

commit 0207864f6893ea9794941e4f3a1a18b644ab03e7
Author: parth13m <parthmadrewar13@gmail.com>
Date:   Wed Aug 6 16:41:36 2025 +0530

    Initial commit with file1 and file2
sysadmin@sysadmin:~/git-lab$
```

Conclusion:

Git helps you manage and track changes in your code locally, while GitHub lets you store your projects online and collaborate with others. Learning both tools is important for teamwork, version control, and working on real-world software projects efficiently.