Name - Dhruv D Vaidya
Batch - T23
Roll no - 117

Experiment 3 : GIT Operations

Aim :

To Perform various GIT operations on local and Remote repositories.

Theory :

Git is a powerful distributed version control system, and it offers a wide range of operations to manage code effectively. These operations can be performed on both **local repositories** (on your computer) and **remote repositories** (hosted on platforms like GitHub, GitLab, Bitbucket).

---

1. Git Repositories: Local vs. Remote

- **Local Repository:** This is the version of the repository on your computer. It contains the working directory, staging area, and the .git folder where Git stores all metadata and history.
- **Remote Repository:** This is a version of the repository hosted on a remote server (like GitHub). It allows multiple developers to collaborate on the same project.

---

2. Git Operations on Local Repositories

2.1. Creating a Local Repository

Initialize a New Repository:

```
git init
```

- This command creates a new `.git` directory in your current folder, initializing it as a Git repository.

Clone an Existing Repository:

```
git clone <repository_url>
```

- This command copies an existing remote repository to your local machine, including its history and branches.

2.2. Tracking Changes

Check the Status of Files:

```
git status
```

- This command shows the current status of the repository, including untracked files, changes to be committed, and changes not staged.

Track New Files (Stage Changes):

```
git add <file>
```

Stage a specific file:

```
git add index.html
```

Stage all changes:

```
git add .
```

Unstage Files:

```
git reset <file>
```

- This command removes a file from the staging area without deleting the changes in the working directory.

2.3. Committing Changes

Commit Staged Changes:

```
git commit -m "Your commit message"
```

- A commit represents a snapshot of your repository at a particular point in time.

Commit with Detailed Message:

```
git commit
```

- This opens the default text editor to write a detailed commit message.

Amend the Last Commit:

```
git commit --amend
```

- This allows you to modify the last commit, either to change the commit message or include new changes.

2.4. Branching and Merging

Create a New Branch:

```
git branch <branch_name>
```

Switch to a Branch:

```
git checkout <branch_name>
```

Create and Switch to a New Branch (Single Command):

```
git checkout -b <branch_name>
```

List All Branches:

```
git branch
```

Merge a Branch into the Current Branch:

```
git merge <branch_name>
```

Delete a Branch (Locally):

```
git branch -d <branch_name>
```

2.5. Undoing Changes

Undo Changes in Working Directory:

```
git checkout -- <file>
```

- This reverts the file to its last committed state.

Unstage Changes (Keep Changes in Working Directory):

```
git reset HEAD <file>
```

Remove the Last Commit (Preserve Changes):

```
git reset --soft HEAD~1
```

Completely Remove the Last Commit (Delete Changes):

```
git reset --hard HEAD~1
```

2.6. Viewing History

View Commit History

```
git log
```

To see a compact version:

```
git log --oneline
```

To view the history of a specific file:

```
git log -- <file>
```

Show Changes in a Commit:

```
git show <commit_hash>
```

---

3. Git Operations on Remote Repositories

3.1. Setting Up Remote Repositories

Add a Remote Repository:

```
git remote add origin <repository_url>
```

- This adds a remote named origin pointing to the given URL.

View Remote Repositories:

```
git remote -v
```

Remove a Remote Repository:

```
git remote remove origin
```

3.2. Pushing Changes to Remote

Push Changes to the Remote Repository:

```
git push origin <branch_name>
```

Example:

```
git push origin main
```

Push All Branches:

```
git push --all origin
```

Force Push (Overwrite Remote History):

```
git push --force
```

- Be careful with this command as it can overwrite history on the remote repository.

3.3. Pulling Changes from Remote

Pull Changes from a Remote Repository:

```
git pull origin <branch_name>
```

- This fetches the changes from the remote repository and merges them into your current branch.

Fetch Changes Without Merging:

```
git fetch origin
```

- This downloads updates from the remote repository without automatically merging them.

3.4. Working with Branches Remotely

Push a New Branch to Remote:

```
git push -u origin <branch_name>
```

Delete a Remote Branch:

```
git push origin --delete <branch_name>
```

Rename a Remote Branch:

```
git push origin :old-branch-name new-branch-name
```

3.5. Handling Merge Conflicts

When multiple people edit the same part of a file, Git may encounter conflicts when merging changes.

Identify Conflicts:

```
git status
```

- **Resolve Conflicts:**
  - Open the conflicted file(s).

Look for conflict markers:

```
<<<<<<< HEAD
(Your changes)
=======
(Incoming changes)
>>>>>>> branch-name
```

  - Edit the file to resolve conflicts, then save.

Mark as Resolved:

```
git add <file>
```

Complete the Merge:

```
git commit
```

---

4. Advanced Git Operations

4.1. Rebasing

Rebase a Branch:

```
git rebase <branch_name>
```

Interactive Rebase (Edit Commits):

```
git rebase -i HEAD~n
```

- This allows you to squash, reorder, or edit commits.

4.2. Stashing Changes

Stash Uncommitted Changes:

```
git stash
```

Apply Stashed Changes:

```
git stash apply
```

List All Stashes:

```
git stash list
```

4.3. Tagging Releases

Create a Tag:

```
git tag v1.0.0
```

Push Tags to Remote:

```
git push origin v1.0.0
```

List Tags:

```
git tag
```

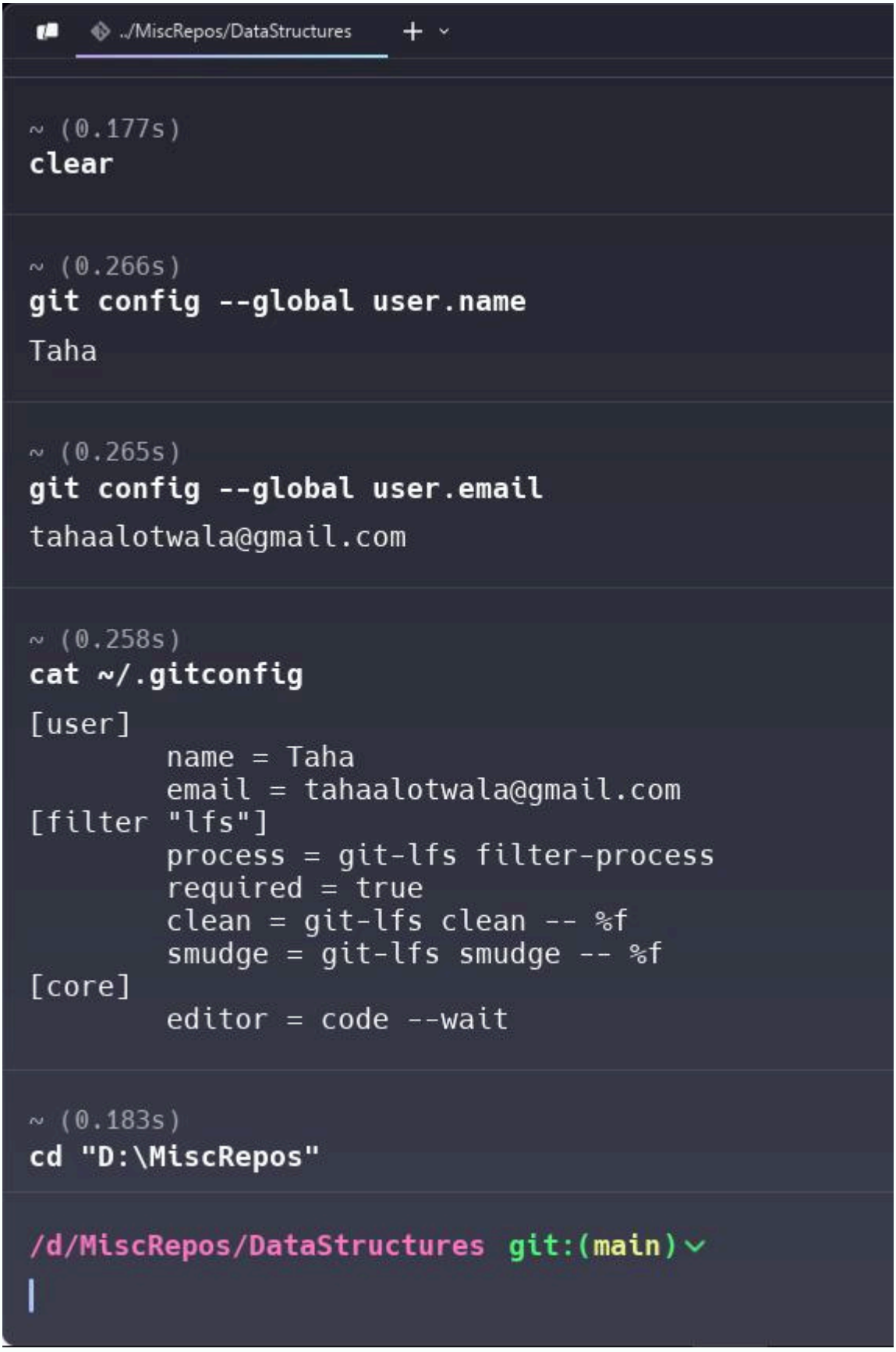
---

5. Summary of Key Git Commands

Operation	Command
Initialize Repository	`git init`
Clone Repository	`git clone <url>`
Check Status	`git status`

Add Files	`git add <file>`
Commit Changes	`git commit -m "message"`
Create Branch	`git branch <branch>`
Switch Branch	`git checkout <branch>`
Merge Branches	`git merge <branch>`
Pull Changes	`git pull origin <branch>`
Push Changes	`git push origin <branch>`
View Commit History	`git log`
Stash Changes	`git stash`
Create Tag	`git tag <tag>`

Screenshots:



```
/d/MiscRepos (0.23s)
mkdir sepm_git_demo

/d/MiscRepos (0.189s)
cd sepm_git_demo/

/d/MiscRepos/sepm_git_demo (0.335s)
git init
Initialized empty Git repository in D:/MiscRepos/sepm_git_demo/.git/

/d/MiscRepos/sepm_git_demo git:(master) (0.237s)
ls -a
./  ../  .git/

/d/MiscRepos/sepm_git_demo git:(master) (0.237s)
ls -al
total 4
drwxr-xr-x 1 Taha 197121 0 Apr  2 10:36 ./
drwxr-xr-x 1 Taha 197121 0 Apr  2 10:36 ../
drwxr-xr-x 1 Taha 197121 0 Apr  2 10:36 .git/
```

```
/d/MiscRepos/sepm_git_demo git:(master) (0.267s)
git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

/d/MiscRepos/sepm_git_demo git:(master) (1m 5.23s)
vi README.md

/d/MiscRepos/sepm_git_demo git:(master)±1 (0.272s)
git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md

nothing added to commit but untracked files present (use "git add" to track)

/d/MiscRepos/sepm_git_demo git:(master)±1 (0.345s)
git add .
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
```

```
/d/MiscRepos/sepm_git_demo git:(master)±1 (0.707s)
git commit -m "initialize repository with a README"
[master (root-commit) 99f1fd7] initialize repository with a README
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

/d/MiscRepos/sepm_git_demo git:(master) (0.334s)
git log
commit 99f1fd7cc12779096dc87b97e9ae520842d9060b (HEAD -> master)
Author: Taha <tahaalotwala@gmail.com>
Date:   Wed Apr 2 10:39:33 2025 +0530

    initialize repository with a README

/d/MiscRepos/sepm_git_demo git:(master) (0.315s)
git log --stat
commit 99f1fd7cc12779096dc87b97e9ae520842d9060b (HEAD -> master)
Author: Taha <tahaalotwala@gmail.com>
Date:   Wed Apr 2 10:39:33 2025 +0530

    initialize repository with a README

 README.md | 1 +
 1 file changed, 1 insertion(+)
```

```
/d/MiscRepos/sepm_git_demo git:(master) (0.328s)
git log --oneline
99f1fd7 (HEAD -> master) initialize repository with a README

/d/MiscRepos/sepm_git_demo git:(master) (0.185s)
cd ../

/d/MiscRepos (1.812s)
git clone "https://github.com/tahaalotwala/DataStructures"
Cloning into 'DataStructures'...
remote: Enumerating objects: 57, done.
remote: Counting objects: 100% (57/57), done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 57 (delta 24), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (57/57), 20.39 KiB | 835.00 KiB/s, done.
Resolving deltas: 100% (24/24), done.

/d/MiscRepos (0.217s)
ls
DataStructures/  sepm_git_demo/  sepm_lab/

/d/MiscRepos (0.187s)
cd DataStructures/
```

```
/d/MiscRepos (0.187s)
cd DataStructures/

/d/MiscRepos/DataStructures git:(main) (0.335s)
ls
1_Stack_using_array.c   5_Circular_Queue.c      7_CircularLL.c     9_BinarySearchTree.c
4_LinearQ_UsingArray.c  6_Singly_Linked_List.c  8_Stack_usingLL.c  README.md

/d/MiscRepos/DataStructures git:(main) (0.28s)
git remote -v
origin  https://github.com/tahaalotwala/DataStructures (fetch)
origin  https://github.com/tahaalotwala/DataStructures (push)

/d/MiscRepos/DataStructures git:(main) (1.078s)
git pull
Already up to date.
```

Conclusion :

Thus, we have successfully studied and performed various GIT operations on local and Remote repositories.