Kishan Wali
T23 123

EXPERIMENT 3

TO PERFORM VARIOUS GIT OPERATIONS ON LOCAL AND REMOTE REPOSITORIES USING GIT CHEAT SHEET

Theory:

Git is a distributed version control system that allows developers to track changes, collaborate, and manage source code efficiently. Git provides numerous commands to handle local and remote repositories.

1. Setting Up Git

Before performing Git operations, configure Git with your details:

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

Verify the configuration:

```
git config --list
```

2. Initializing a Git Repository

To create a new Git repository:

```
git init
```

This initializes a new repository in the current directory.

3. Cloning a Repository

To clone a remote repository:

```
git clone <repository_url>
```

Example:

```
git clone https://github.com/your-username/repository.git
```

4. Staging and Committing Changes

- To check the status of the working directory:
- git status
- To add files to the staging area:
- git add <file_name>

or to add all changes:

```
git add .
```

- To commit changes with a message:
- git commit -m "Your commit message"

5. Viewing Commit History
To view commit logs:

git log
For a compact version:
git log --oneline
6. Branching in Git
- To create a new branch:
- git branch <branch_name>
- To switch to another branch:
- git checkout <branch_name>
- To create and switch to a new branch simultaneously: git checkout -b <branch_name>
- To view all branches:
- git branch

7. Merging Branches
- First, switch to the main branch:
- git checkout main
- Merge a branch into the main branch:
- git merge <branch_name>

8. Pushing Changes to Remote Repository
- To push changes to GitHub:
- git push origin <branch_name>
- If pushing for the first time:
- git push --set-upstream origin <branch_name> 9. Pulling Changes from Remote Repository To fetch and merge changes from a remote repository: git pull origin <branch_name>

10. Handling Merge Conflicts
If a merge conflict occurs:

1. Open conflicting files and resolve issues manually. 2. Add resolved files to the staging area:
3. git add <file_name>
4. Commit the resolved changes:
5. git commit -m "Resolved merge conflict"

11. Undoing Changes

- To undo changes before staging:
- git checkout -- <file_name>
- To unstage a file:
- git reset HEAD <file_name>

- To revert the last commit:
- git revert HEAD

12. Deleting a Branch
    - To delete a local branch:
    - git branch -d <branch_name>
    - To delete a remote branch:
    - git push origin --delete <branch_name>

13. Creating and Using a .gitignore File

A .gitignore file is used to ignore specific files or directories: echo "node_modules/" >> .gitignore

git add .gitignore

git commit -m "Added .gitignore file"

14. Checking Differences in Files
    - To compare working directory changes:
    - git diff
    - To compare staged changes:
    - git diff --staged

15. Stashing Changes

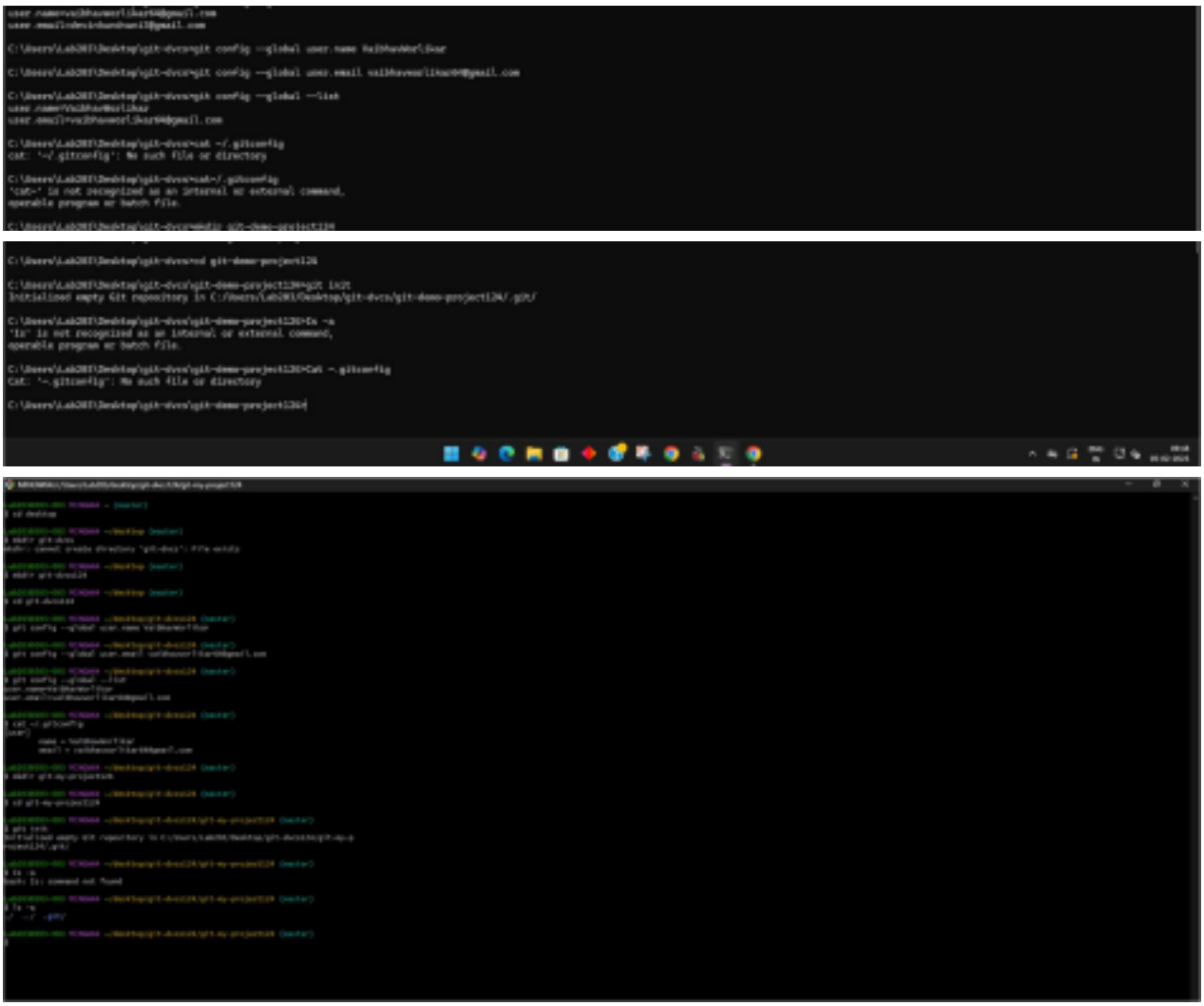
To temporarily save uncommitted changes:

git stash

To apply the stashed changes:

git stash apply

Output:



Conclusion

This experiment demonstrated various Git operations, including repository initialization, branching, merging, pushing, pulling, and resolving conflicts. These commands help in efficient version control and collaboration in software development projects.