Experiment No.3
To Perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet
Date of Performance:
Date of Submission:

Aim: To Perform various GIT operations on local and Remote repositories using GIT Cheat-Sheet

Objective: Objective is to acquire proficiency in common Git commands and workflows, enabling efficient version control, collaboration, and project management in software development projects, both locally and across distributed teams

Theory:

Git Commands

For setup:

- git config --global user.name "Enter User name of Github Account"
- git config --global user.email "Enter email of Github Account"

For Initialization:

- git init : Initialize an existing Directory as a Git Repository.
- git clone [url] : Retrive an entire repository from a hosted location via URL (Paste HTTPS OR SSH key From your gitHub)

Stage and SnapShot :

(Following commands works with respect to staging area)

- git status : show modified files in working directory, staged for your next commit
- git add [file] : add a file to a staging area
- git reset [file] : (get file back from staging area)unstage a file while retaining the changes in working directory
- git diff : diff of what is changed but not staged
- git diff --staged : diff of what is staged but not yet commited
- git commit -m "[type a message]" : commit your staged content as a new commit snapshot

Branch & merge :

(work in branches, changing context, and integrating changes)

- git branch : list your branches. a * means currently active branch
- git branch [branch-name] : create a new branch at the current commit
- git checkout : switch to another branch and check it out into your working directory
- git merge [branch] : merge the specified branch's history into the current one
- git log : show all commits in the current branch's history

Inspect & compare

- git log : show the commit history for the currently active branch
- git log : branchB..branchA : show the commits on branchA that are not on branchB
- git log --follow [file] : show the commits that changed file, even across renames
- git diff branchB...branchA : show the diff of what is in branchA that is not in branchB
- git show [SHA] : show any object in Git in human-readable format

Share & update :

(Retrieving updates from another repository and updating local repos)

- git remote add [alias] [url] : add a git URL as an alias
- git fetch [alias] : fetch down all the branches from that Git remote
- git merge [alias]/[branch] : merge a remote branch into your current branch to bring it up to date
- git push [alias] [branch] : Transmit local branch commits to the remote repository branch
- git pull : fetch and merge any commits from the tracking remote branch

Tracking path changes :

(Versioning file removes and path changes)

- git rm [file] : delete the file from project and stage the removal for commit
- git mv [existing-path] [new-path] : change an existing file path and stage the move
- git log --stat -M : show all commit logs with indication of any paths that moved

Rewrite history

(Rewriting branches, updating commits and clearing history)

- git rebase [branch] : apply any commits of current branch ahead of specified one
- git reset --hard [commit] : clear staging area, rewrite working tree from specified commit
- git stash : Save modified and staged changes
- git stash list : list stack-order of stashed file changes
- git stash pop : write working from top of stash stack
- git stash drop : discard the changes from top of stash stack

Conclusion:

Q1. How to retrieve an entire repository from a hosted location via URL?

To retrieve an entire repository from a hosted location via URL, you can use Git's clone command in your terminal or command prompt. Begin by navigating to the directory where you want the repository to be cloned. Then, use the command `git clone` followed by the URL of the repository you wish to retrieve. For example, `git clone https://github.com/username/repository.git`. This command will create a new directory with the repository's contents, including all branches and commit history, in your current location. Ensure you have Git installed and configured on your system, and that you have the necessary permissions to access the repository.

Q2. How to change an existing file path?

To change an existing file path, you'll need to use the appropriate functions provided by your operating system or programming language. In general, you would first identify the current file path and then use methods like `os.rename()` in Python or `mv` command in Unix-based systems to move or rename the file to the new path. Ensure that you have the necessary permissions to modify the file, and be cautious to avoid overwriting important data. Remember to handle any errors that might occur during the process, such as file not found or permission issues, to ensure a smooth transition to the new file path.