Introduction to Git and Github

By

Adnan Kamal Chowdhury

ID: 2104010202297

CSE306: Software Engineering and Information System Design Lab

Instructor:

MD. Tamim Hossain

Lecturer

Department of Computer Science and Engineering

Premier University

_____

Signature

Department of Computer Science and Engineering

Premier University

Chattogram-4000, Bangladesh

13 November,2023

Introduction:

Git: Git is a open-source distributed version control system designed to track changes in source code during software development. It provides a way for multiple developers to collaborate on projects, managing and merging code changes efficiently. It is designed to handle everything from small to very large projects with speed and efficiency. With Git, everytime if anyone commit, or save the state of the project, Git basically takes a picture of what all files look like at that moment and stores a reference to that snapshot. So it's a powerful tool that is used to keep track of code changes and collaborate on code.

Github: Github is a platform where developers collaborate on projects using Git, which is a version control system. It allows multiple people to work on the same codebase simultaneously, tracking changes, managing different versions and merging contributions seamlessly. Github provides tools for code hosting, reviewing changes, managing issues and facilitating collaborations among developers and teams. It's widely used in the software development

community for open-source projects, team collaborations, and individual coding projects.

Activities:

Activity 1: Downloaded the git and installed it in my personal computer.

Activity 2: configured git with my username and email using the 'git config' command.

Activity 3: created my first Git repository using the 'git init' command in my project directory.

Activity 4: Set the default branch and it's name by using 'git config' command.

command: git config --global init.default branch main

Activity 5: Checked the condition/status of my repository using the command 'git status'.

Activity 6: Created a file 'test1.txt' and that file is added to git, so that git can track that file.

Command: git add test1.txt

Checked if the file is tracked by Git using the previous command 'git status'.

Activity 7: Used 'git commit' command to create a snapshot of the changes made to that in Git repository. By this command, I permanantly saved the changes in the project history.

command: git commit -m " first file "test1" added "

Activity 8: Deleted an unnecessary file from my Git repository by the following command:

'git rm 'test2.txt'.

Activity 9: Set up my Github account in 'github.com' and created a repository named as 'Lab-1'.

Activity 10: Established a remote connection between my github account and local computer repository using this command:

'git remote add origin https://github.com/Adnan Chowdhury14/Lab-1.git '

Then pushed my existing repository to the github cloud by these command:

'git branch -M main '

'git push -u origin main'

Git Cheat Sheet:

1. git config --global user.name "[firstname lastname]"
   - set a name that is identifiable for credit when review version history.

2. git config --global user.email "[valid-email]"
   - set an email address that will be associated with each history marker.

3. git config --global color.ui auto
   - set automatic command line coloring for Git for easy reviewing.

4. git init
   - intialize an existing directory as a Git repository.

5. git clone [url]
   - retriere an entire repository from a hosted location via URL.

6. git status
   - show modified files in working directory, staged for next commit

7. git add [file]
   - add a file as it looks now to your next commit (stage)

8. git reset [file]
   - unstage a file while retaining the changes in working directory.

9. git diff
   - diff of what is changed but not staged.

10. git diff --staged
   - diff of what is staged but not yet committed.

11. git commit -m "[descriptive message]"
   - commit the staged content as a new commit snapshot

12. git branch
   - list your branches. a * will appear next to the currently active branch.

13. git branch [branch-name]
   - create a new branch at the current commit.

14. git checkout
   - switch to another branch and check it out into your working directory.

15. git merge [branch]
   - merge the specified branch's history into the current one.

16. git log
   - show all commits in the current branch's history.

17. git log branchB..branchA

- show the commits on branchA that are not on branchB.

18. git log --follow[file]

- show the commits that changed file, even across renames.

19. git rm [file]

- delete the file from project and stage the removal for commit.

20. git log --stat -M

- show all commit logs with indication of any paths that moved.

21. git remote add [alias] [url]

- add a git URL as an alias.

22. git fetch [alias]

- fetch down all the branches from the Git remote.

23. git merge [alias]/[branch]

- merge a remote branch into your current branch to bring it up to date.

24. git push [alias] [branch]

- Transmit local branch commits to the remote repository branch.

25. git pull

- fetch and merge any commits from the tracking remote branch.

Discussion: The introduction to Git and Github provided a fundamental understanding of version control and collaborative development. Through a series of introductory activities. I gained hands-on experience with the basic functionalities of Git as a version control system and Github as a collaborative platform. Setting up Git locally enabled tracking changes, while creating repositories, adding files and committing changes illustrated version control concepts. It Integrating Git with Github highlighted collaborative features, allowing sharing and syncing of code. This foundational experience showcased the importance of Git/Github in modern development workflows, laying the groundwork for future collaborative coding.