Premier University Chattogram

Report on

Introduction to Git and Github

Course Title : Software Engineering & Information System Design Lab

Course Code : CSE – 306

Report No. : 01

Submitted to :

MD. Tamim Hossain
Lecturer
Department of CSE, PUC

Submitted By :

Shomitra Dey Dipon
ID : 2104010202301 , Sec : D
Semester : 5th , Batch : 40th

Submission date : 18th November,2023

Department of CSE

Premier University, Chattogram

Abstract:

This lab report provides an overview of the research conducted to introduce Git & Github, essential tools in modern software development. The report aims to familiarize the reader with the purpose of using version control system and collaborative platforms for managing code.

Introduction:

Git, a distributed version control system, and GitHub, a web based platform for hosting Git repositoriers, have become integral in modern development workflows. It allows multiple developers to collaborate on a project by managing and merging different versions of database. The hypothesis is that adopting Git and GitHub will lead to enchanced code management, and project efficiency.

Materials:

- Computer.
- Git .
- GitHub account.
- Text editor.

Activity:

1. Create a git repo:

i) Create a directory you want to set as your repsitory in a location:

- $ mkdir myFirstRepo: This command creates a new directory named "MyFirstRepo".
- cd myFirstRepo: This command changes the current directory "myFirstRepo".

ii) Initialize the directory as a repository:

```
$ git init
$ git config --global init.defaultBranch main
$ git branch -m main
```

iii) Use config to add name & email:

```
$ git config --global user.name "soumitra"
$ git config --global user.email "soumitradev532
                                  @gmail.com"
```

2. Create a txt script that prints out "Helloworld" and run it in the terminal:

i) Create a txt script in directory:

First.txt

ii) Inside the file write the text "Hello World!"

3. Change First.txt and run it on the terminal, then add it, commit it and git status and git diff:

i) Add a file to staging:

```
$ git status (shows which state the file are in.)
$ git add First.txt
$ git status
```

ii) Commits files in staging:

```
$ git commit -m "File added"
$ git log (to see what our commits look like.)
$ git status
```

iii) Change, save and exit out of the text file:

```
$ git diff (shows differences between commited
                file & current staged file.)
```

iv) Commit it again:

```
$ git commit -m "Edited current file"
$ git log
$ git status
```

Git command list:

1) git config --global user.name (set the name that will be attached to user commits and tags.)

2) git config --global user.email "xyz@gmail.com" (set the email address that will be attached to user commits and tags.)

3) git status (Displays the status of working directory. Option included new, staged & modified files.)

4) git add [file] (Add a file to the staging area.)

5) git diff [file] (shows changes between working directory & staging area.)

6) git checkout [file] (Discard changes in working directory. The operation is unrecoverable.)

7) git commit (create a new commit from changes added to the staging area. The commit have a message)

8) git rm [file] (Removes file from working directory and staging area.)

9) git stash (Put current changes in your working directory into stash for later use.)

10) git stash pap (Apply stored stash content into working directory and clean trash.)

11) git stash drop (Delete a specific stash from all previous stashes.)

12) git branch [-a] (List all local branches in repository. show all branches.)

13) git branch [branch_name] (create new branch, referencing the current HEAD.)

14) git checkout [-b] [branch_name] (switch working directory to the specific branch. Git will create branch if there is none.)

15) git merge [branch_name] (Joins specified branch into current branch.)

16) git branch -d [branch_name] (Removes selected branch.)

17) git log [-n count] (List commit history of current branch.)

18) git log ref.. (List commits that are repre -sented on the current branch and not merged into ref.)

19) git log ..ref (List commits that are present on ref and not merged into current branch.)

20) git tag (List all tags.)

21) git tag -a [name] [commit sha] (create a tag object named "name" for current commit.

22) git tag -d [name] (Remove a tag from local repository.)

23) git fetch [remote] (Fetch changes from the remote, but not update tracking branches.)

24) git pull [remote] (Fetch changes from the remote and merge current branch with it's upsteam.)

25) git push -u [remote] [branch] (Push local branch to remote repository)

Discussion:

When I first started using Git & GitHub, I ran into a few common problems. One big challenge was trying to understand all the Git commands like commit, push, pull and merge. Figering out how to use branches and fix conflicks also felt a bit confusing at first. Understanding these early issues and finding ways to make things easier is important for using these tools better.

Conclusion:

In conclusion, my exploration of Git and GitHub has been both challenging and rewarding. Despite initial difficulties, these tools significantly enhance code management and collaboration. Git and GitHub are now essential for efficient software development.