GIT and GITHUB

Monday, January 6, 2025 11:36 AM

The Relationship Between Git and GitHub

- **Git** is the underlying technology that **GitHub** uses. While Git handles the version control aspects locally on your machine, **GitHub** provides a cloud-based platform to host and manage your repositories and collaborate with others.
- **Git** is a tool for version control, and **GitHub** is a hosting platform for your Git repositories.

How to link (local to remote project) git repositories on browser to the local machine (via CMD):

1. Clone the repository (copy the link)
2. In cmd type - git clone <link you have copied>
3. cd <name of the repository which link u have cloned> //e.g. cd gitCourse
4.for verfying : git remote -v //it will give the fetch, push link of the remote repository

Push and Fetch Operation.

PUSH- when we need to update our changes done on local to the remote repository.
FETCH (PULL)- When we need to retrieve the changes done by others or anyone from the Central (Remote) Repo to our local repo
Adding -h to the end of a git command will display the available parameters.

COMMIT

A set of save repository, never lost, has a hash key, always same
Command:
git add . - add all the file changes to the staging area
git commit - allows us to create commit e.g.- git commit -m "Initial Commit" - m is for giving comments.
git log- gives the overview of all the commits like hash key of commit, branch of the commit.

STAGING

Marking files them to be committed. Sometimes we want some files to be committed then we use specific file name e.g. - git add file1.text file3.txt file5.txt
git reset <filenames> - remove the files from staging area.
Git status- gives overview of staged and untracked files.

Unstaged Files:

- **Unstaged files** are files that have been **modified** in your working directory but haven't been added to the **staging area** yet.
- These changes are **not** ready to be committed to the repository; they are still local to your working directory.
- Essentially, these files are **not tracked** for the next commit.

Staged Files:

- **Staged files** are files that have been **added** to the staging area using the git add command.
- Staging means you've told Git that you want to include these changes in the **next commit**.
- Essentially, the staging area is a preview of the changes you want to commit. Once staged, the changes are ready to be committed.

If one or more files are in staging then we can do commit.

BRANCH

A series of commits that can be manipulated independently
(e.g. a scenario in which we are working on Feature A and suddenly we have to switch to Feature B then back to A)

Master Branch- Code that is currently most up to date.
All the branches are merged into main branch once everything is done.
git branch- displays all the branches
Git branch featureBranch - add new branch
Creating new file in master branch and featureBranch
Echo master branch >newFileMaster.txt (after this do add and commit)
Git checkout <branchName u want to switch to different branch) e.g. - git checkout featureBranch
Echo feature branch >newFileFeautureBranch.txt (add and commit)

GIT SHOW

Git show -information about a specific command(we need hash value of that commit - we get hash value from git log)
Git show <commit hash value > - all the info when the file name changes or any updates.
Git show --name-only <hashvalue> - it will give the name of all the files in that commit.
To **exit the loop** and return to your command prompt, you can do the following:
1.**q** is the command to **quit** the pager (like less or more) in Git.
2.Use git show with --no-pager:
git show --no-pager <hash value>

GIT REFLOG

Gives the information about all the changes that happened on the project , including all commits on all the branches

PUSH and PULL

Creating a separate repository on our local machine based on the same project (two same projects pointing to the same central repository, both are independent)

1. Copy the repo link
2. Make sure you are out of previous repo project.. Then git clone <link> but u have to give new name to the repo.
   Git clone <link> newRepoName
3. cd newRepoName
4. git log
5. echo testCommit > testCommit.txt
6. git add .
7. git commit -m "Test Commit"
8. git push
9. Cd ../GitCourse - How to move to other folder in CM
10. git pull - GitCourse repo wii retrieve all the changes from the newRepository

UNDO/REDO

Option A
REVERT- create a new commit that undoes the changes of a certain limit
Git revert <hash value of the commit we want to revert>
You are stuck in the editor, and you need to exit and either save or discard your changes.
Here's how to exit **vim** (or similar editors) and return to the command prompt:

- In **vim**: :q! to quit without saving or :wq to save and quit.
- In **nano**: Ctrl + X, then N to discard or Y to save and exit.

Option B
RESET- Resets the current branch to show only commits upto the one you indicate. All the subsequent commits are disregarded.
Git reset --soft <hash> -disregards subsequent commits but keeps the changes they have introduced (default)
Git reset --hard <hash> -disregards subsequent commits including changes if no modifier is specified.

MERGING BRANCHES

1. Go the the repo GitCourse
2. Git merge featureBranch
3. Git log
4. We will see a new commit that was created when the branches were merged

Merge conflicts can happen whenever the same file was modified in different ways on 2 branches and we are trying to merge

Resolving merge conflicts:

PULL REQUESTS

It is a way for one contributor to have a look at another contributors code, give opinions and reviews and approve or reject the change.

It is basically a request to merge a branch into another branch

1 Git push -f :-f tells git that the local branch is correct and that is should force the override the remote one (because we have reverted some commits)

Pushing the local branch on the central Repo :

2 Git push -u origin featureBranch //-u is the remote Repo

Now as both the branches are on central repo now we can call a pull request

3 Add one file abd commit to featureBranch as newCommit

4 Go to github, click on project link->switch to featureBranch-> click new pull request-> click create pull request-> Here we can see the code changes that were introduced by this PR, reviewers can add comments or suggestion at the line where the comment applies. We can add reviewers to the PR in the top right corner i.e. conversation tab.

You can always add new changes to the PR by uploading new commits to the branch.

5 Once the PR is approved , PR can be merged by cling MERGE PULL REQUEST.

Some more commands

1 git init -h : creates a new local repository in a folder
2 git remote add -h : adds a new remote repo to the project
3 git diff sourceBranch targetBranch : target branch shows you the difference between 2 branches
4 git tag -h : creates a snapshot of the project at a certain time
5. Git stash -h : moves the changes of the project into a stash
6. 6 git stash pop -h : retrieves changes from stash and reapplies them
7. Git rebase -h: takes commits that have happened in the original branch and applies them into the current branch before the current changes
8. Git clean -h : remove the untrack files