Git & GitHub

Git

- Git is a popular **version control system** i.e. the practice of tracking and managing changes to software code.
- Git was created by Linus Torvalds in 2005.
- Git is used for:
  - Tracking code changes.
  - Tracking who made changes.
  - Code collaboration.
- Git does the following:
  - Manage projects with Repositories.
  - Clone a project to local machine.
  - Control and track changes with Stagging and Committing.
  - Branching and merging to work on different parts and versions of a project.
  - Pull out the latest codes.
  - Push the local changes.

Installing Git

- Use the following page to download git: Download Git
- Verify git is installed successfully by entering following command in **cmd:** *git --version*
- After you enter this command, you should see a message like as follows: ***git version X.Y***
- If you get an error message ***"unable to recognize git",*** then git is not installed properly.

Commands for Configuring Git

- *git config --global user.name "YOUR_USERNAME"*
- *git config --global user.mail "YOUR_EMAIL"*

- Before you configure make sure that you have an account in **GitHub** or create one on github.com
- After you have your account replace **"YOUR_USERNAME" & "YOUR_EMAIL"** with your credentials.
- Here we used ***--global*** to make sure that the configuration applies to all repositories in our computer.
- If we simply want to set the configuration to the current repository, then remove ***'--global'***.
- Without Configuring git if we try to push projects to GitHub, they may be pushed successfully but your contributions are not displayed in your profile.

Stagging & Committing in Git

- In **Stagging Environment**, Stagged files are there i.e. the files that are ready to be committed.
- In **Committing Environment**, Commits takes place. Commits are just like save points; they keep track of our work along with a provided required message.

git help

- If we encounter any error while working with any git command, then we can enter the below commands:
  - **git command -help**: to see the available options of specific command.
  - **git help --all**: to see the available options for all commands.

Git Branches

- In Git, a branch is a new/separate version of main repository.
- Branches allow us to work on different parts of a project without impacting the main branch.

- When the work is done, a branch can be merged with the main branch.
- We can easily switch between different branches.
- To **create** a new branch in git, enter the below command:
  - git branch BRANCH_NAME
- To **switch** to an Existing branch in git, enter the below command:
  - git checkout BRANCH_NAME
- To **create and switch** to a new branch, enter the below command:
  - git checkout -b BRANCH_NAME
- To **merge an Existing branch** to main branch, use the below command:
  - git merge BRANCH_NAME
- To **remove an Existing branch**, enter the below command:
  - git branch -d BRANCH_NAME
- The above command will delete the branch locally but in GitHub we can still see the deleted branch, we need to delete the remote branch as well using the following command:
  - git push origin --delete BRANCH_NAME

git pull

- Let us imagine a scenario where you along with your team are working on a project and everyone has to complete their own piece of work in slots, now whenever you start working on the project by cloning it to your local machine you need to get the updated files every time when you start working and for this purpose we can use **git pull.**
- **Git pull** is a combination of two commands:
  - **fetch:** It gets all the changed history of a branch/repository.
  - **merge:** It combines the branch with present branch.
- To implement the above scenario of working on a project and detecting changes we need to use the following:
  - fetch
  - merge
- Using git pull these two steps can be implemented as a single step:
  - **git pull == git fetch + git merge**

git status

- It displays the status of the repository including tracked/untracked files and changes staged for commit.

git log

- This command displays the commit history of the repository.

git diff

- This command shows the difference between changes in the working directory and staging area of recent commit.

Git Fork

- As the heart of git is collaboration, git does not allow us to add code to someone else's repository without access.
- A fork is a copy of a repository, and it is useful when we want to contribute to someone else's code.
- **Fork is not a git command,** but it is offered in GitHub.
- Now we have our own fork, but only on GitHub and we also want a clone on our local computer.
- A **clone** is a complete copy of a repository, and it can be done by getting the repository URL.
- Command to clone:
  git clone https://username@github.com/username/repository.git
- After cloning, add a few changes and try to push the changes to git by using the ***"git push origin"*** command.
- This will create a pull request in the main repository and the owner can accept the pull request or reject the pull request to implement changes.

Git .gitignore

- When sharing our code with others, there may be some files which we don't want to share.
- In git we can specify which files to be **ignored** by using a **.gitignore file.**
- Git will not track the files specified in **.gitignore file.**
- To create a **.gitignore** file, enter the following command:
  touch .gitignore
- In this file we can directly specify the filename, extension of filenames, directories, etc to be ignored.
- For example, if I want to ignore a file named as "index.html" then simply by specifying the filename is enough.

git revert

- Revert is the command that is used to take a previous commit and add it as a new commit.
- First, we find the logs by using the below command: ***git log --oneline***
- This will display all commits line by line with a 7 digit commit hashkey.
- We simply want to revert to the previous commit, then use the below command: ***git revert HEAD --no-edit***
- Here ***"--no-edit"*** is used to have the same commit message of the previous one.
- If you want to revert to some xth previous commit, then use below command: ***git revert HEAD~x***

Git Reset

- Reset is the command that we use when we want to move the repository to the previous commit.
- Enter this command to get all commits: ***git log --oneline***
- Get the 7-DIGIT_COMMIT_VALUE for your commit that you want to reset.

- Enter the following command: ***git reset 7-DIGIT_COMMIT_VALUE.***
- Execute the commands below to reflect your changes in the repository:
  - git commit -m "COMMIT VALUE"
  - git push -u origin BRANCH_NAME

GitHub

- GitHub hosts Git repositories and provides developers with tools to ship better code through command line features, issues (threaded discussions), pull requests, code review, or the use of a collection of free and for-purchase apps in the GitHub Marketplace.
- GitHub makes tools that use Git. GitHub is not the same as Git.
- GitHub is the largest host of source code.
- GitHub has been owned by Microsoft since 2018.

Pushing Files to GitHub

- We can push our files to GitHub in two ways:
  - Drag and Drop Method
  - Command Based Method

- If your project directory / folder has a very limited number of files, then you can follow the following steps for Drag and Drop Method:
  - open **github.com** in web browser and sign-in to it.
  - click create repository and name it as per your wish.
  - If interested add a "readme.md" file as it is optional.
  - Click create repository and then directly drag and drop your project folder.
  - Wait for some time until all files are uploaded.
  - Edit the commit value or leave it default.
  - click commit changes to save all the files in your repository.
  - verify your files are uploaded or not.

Note-1

- **"readme.md"** file is a file that explains about your files that you have added in your repository, here you can add all the information about project files, etc.
- The above method of **"drag and drop"** works only when the file count limit is below 100.
- If your project directory / folder has more than 100 files, then follow below steps:
    - open command prompt or terminal.
    - Navigate to the project directory using "cd" command.
    - cd C:\Users\DELL\Desktop\SDP\Portfolio
    - After navigating enter below commands:
        - ***git init***
        - ***git add .***
        - ***git commit -m "SOME COMMIT NAME"***
        - ***git remote add origin https://github.com/username/repository***
        - ***git push -u origin master***
    - After executing all the above commands without any errors, your folder is added to GitHub.

- The default branch in our local machine is **master only** and that's the reason why we use this command to push: **git push -u origin master.**
- If we want to push directly to default **main branch** available in github repository, then use the below commands:
    - ***git branch -m master main***
    - ***git push -u origin main --force***
- The above two commands are used to change our local branch to main from master to push the files.

Note-2

- Here Portfolio is the Project folder which is added by navigating to that folder using following cd command:
  cd C:\Users\DELL\Desktop\SDP\Portfolio
- **"git init"** is used to initialize a new repository in your project folder.
- **"git init"** creates a .git file which consists of all information about commits and file metadata, etc.
- **"git add ."** is used to add all the files to the repository.
- **"git commit -m "SOME COMMIT NAME"** is used to create a snapshot of changes.
- **"git remote add origin https://github.com/username/repository"** is used to connect to our repository which we created.
- To get this **"https://github.com/username/repository"** simply open the repository and copy the address bar URL of your browser.
- **"git push -u origin master"** is used to push the folder to GitHub repository.

Note-3

- The above execution of commands with create a **"master"** branch in repository and your folder files are added to that master folder.
- By default, GitHub repository consists of a **"main"** branch.
- If you want to make your **"master"** branch as default, then follow below steps:
  - Click settings button, you will be redirected to settings.
  - Under the Default branch you will see a bidirectional arrow symbol "<=>" somewhat like this.
  - Click on it and you can switch to the branch you want.
  - Now when you open your instead of "main" branch your "master" branch will appear as default.
  - If you want to delete "main" branch, then you can find a button as branches with branch count.
  - Click on it and you can delete the "main" branch.

Steps to Delete / Clear initialized git from folder

- Navigate to the project directory using "cd" command:
  cd C:\Users\DELL\Desktop\SDP\Portfolio
- Enter anyone of below command based on your utility:
  - **Remove-Item -Path .git -Recurse -Force** (For Windows PowerShell)
  - **rmdir /s /q .git** (For Command Prompt)
  - **rm -rf .git** (For bash)
- This command is used to remove the initialized git repository from our folder.
- After that open GitHub.com and go to settings of your project and scroll down to last, you will find a button to delete repository.

Steps to update new changes to an existing Repository

- Enter **"git status"**, this will show the updated changes.
- Enter **"git add ."** to add the changes into repository.
- Enter **"git commit -m "SOME COMMIT"** to save all the changes.
- Enter **"git push -u origin master"** to update the repository.

Steps to handle deleted files

Imagine a scenario where you have deleted a couple of files from your local repository, but your changes are not reflected in GitHub repository and the deleted files are still available in your repository, to ensure that deleted files are also removed from repository use below commands:

git status

git add -u

git commit -m "Removed deleted files from the repository"

git push origin main

Steps to Work on Other Person's Repositories (Collaboration)

1. Open the Repository
    - Open the other person's repository in your account.
    - Click on the Fork button (available at the top-right corner).
2. Fork the Repository
    - Forking creates a copy of the repository in your account.
3. Clone the Forked Repository
    - Open the terminal and clone your forked repository locally:

      git clone <your-forked-repository-url>
4. Navigate to the Project Directory
    - Use the following command to navigate to the repository folder:

      cd <repository-name>
5. Add Upstream Remote
    - Add the original repository as an upstream remote to sync future updates:

      git remote add upstream <original-repository-url>
6. Make and Commit Changes
    - Make the required changes and commit them:

      git add .

      git commit -m "update other person's code"

      git push -u origin main

7. Submit a Pull Request
    - To reflect the changes in the original repository:
        - Go to your forked repository.
        - Click on Contribute.
        - Click on Open Pull Request.
        - Provide a title and description for your pull request.
        - Click Create Pull Request.
8. Approval Process
    - Once the pull request is created, the repository owner will see it in their repository.
    - The owner can review and merge the changes by clicking on Merge Pull Request. If accepted, your changes will be reflected in their repository.

Steps to Update a Repository to a Particular nth Commit

1. View Commit Log History
    - Use the following command to display the commit history:

      git log --oneline
2. Checkout to the nth Commit
    - Move the code to the state it was in at a particular commit:

      git checkout <commit-hash-of-nth-commit>
3. Create a New Branch from the nth Commit
    - Create a new branch to work from the nth commit:

      git checkout -b fix-nth-commit

4. Make and Commit Changes
    - Edit the files and commit the changes:

      git add .

      git commit -m "Fix nth commit"
5. Switch Back to the Main Branch
    - Move to the main branch:

      git checkout main
6. Merge Changes
    - Merge the changes from your new branch:

      git merge fix-nth-commit
7. Push the Changes
    - Push the updated main branch to the repository:

      git push -u origin main

Disclaimer

- The following notes are general summaries and overviews of the topics discussed.
- These notes are not exhaustive and do not cover all aspects of the subject matter.
- The information provided herein is intended for educational purposes only and should not be used as a substitute for professional advice, detailed study, or official course materials.

References

For more detailed information, please refer to the following resources:

Reference 1: Complete Git & GitHub Tutorial

Reference 2: Complete Git & GitHub Tutorial

Reference 3: Git & GitHub Cheat Sheet

Reference 4: Git & GitHub Most Asked Questions

Reference 5: Top Git & GitHub Interview Questions

Prepared By: ***Reddy Venkat Kalyan***
