**Git:**

1. Git is a version control tool, not a specific programming language but it is useful for bigger and sophisticated web applications.

2. It is a Command Line tool that allows us to keep track of changes that we make to code.

3. It makes us synchronize the code between different people means that other people can access the code.

4. The other person can access the repository, update the code, and save it back to the server then we have the most updated version of the code.

5. To update the code we **"push"** the code into the server and to access the code we

**"pull"** the code from the server to get the latest version of the code.

6. Git allows us to test the changes in the code without removing access to the original code.

7. Git allows us to change the code on a separate branch and if we are ok with the changes, git allows us to merge those changes back into the original version of the code.

8. In git we can revert and get the previous code version instead.

9. Git is an online platform where we can store the code and download it from anywhere.

10."Github" is a website that stores the Git repositories, whereas a folder holds a whole bunch of code and files related to our code.

11. first sign in into your GitHub account -> Create a new Github repository -> https://github.com/new

-> Give the repository name -> Description is optional in the repository -> Select the public or private for the repository.

-> Click on Create Repository.

12. The **"git clone URL"** of the repository is a command used to take a repository from the internet and download it onto our computer.

13. **"ls"** means list and if we pass the command it returns the current files present inside the directory.

14. To change the directory we use the **"cd"** command.

15. **"git add filename"** is a command that tells the git to add the file as one to track the next time I save.

16."git commit" is a command that tells the git repository that I want to save a snapshot of the current state of the repository

and to keep track of any of the changes that have been made to files that I have added.

17. **git commit -m "message",** the message is known as a commit message, or what changes you've made in the most recent commit. It is just a note of message to ourselves -- such that we can refer back to all our commit messages.

for example, git commit -m "Add the line" means that we added some lines.

18. Git keeps track of changes how many lines are added/inserted or how many lines have been deleted/removed.

19. Till now we have made the changes only to the local computer but not the online repository of the GitHub.

20. **"git status"** tells us what is currently happening inside my repository.

21. The **"git push"** command helps us publish our local files into the repository by actually pushing them up to the server, and then pushing them to GitHub.

22. If we change the files of the local computer, the modified changes cannot be saved to the online GitHub repository.

23. To save the changes we use **"git add filename" -> it is for a single file.**

24. If we have to **modify multiple files we use a command called git command -am "Add some message"** means that commit all the files that have been changed.

This command is a combination of the add and message of the git commands. The changes are modified in the local computer but not the repository.

To change in the repository use **"git push"** so all the changes are updated to the online GitHub also.

25. check the status of the git whether the git is modified or not with the help of the **"git status"** command.

26. The git pull is opposite to the git push. git pull takes changes present in the changes that currently exist on the GitHub, and pulls the recent changes of the repository to the local version of the repository.

27. If 2 make changes to the same code, then we have a conflict called **"Megre Conflict".**

28. It shows an error message called

**CONFLICT(content): Merge conflict in filename**

**Automatic merge failed; fix conflicts and then commit the result**.

29. This error shows 2 phases: one error on the local computer and the other is changed on the online GitHub repository.

for eg:

a) On my Local computer I have:

1.<html>

2.<head>

3.<title>Conflicting of the git</title></head>

4.<body>

5.<h1>Welcome to the website!!</h1>(here I have another exclamatory mark that is not present in the repository)

6.<h2>Hello!</h2></body>

7.</html>

b)In the online repository I changed to:

1.<html>

2.<head>

3.<title>Conflicting of the git</title></head>

4.<body>

5.<h1 style="color:blue">Welcome to the website!</h1>(here I am adding css property that is not in the local computer)

6.<h2>Hello!</h2></body>

7.</html>

c) From the above we are changing the code on the same line 5. The conflict that is shown when we run the git pull command: Merge Conflict.

It shows the error like this:

1.<html>

2.<head>

3.<title>Conflicting of the git</title></head>

4.<body>

**<<<<<<HEAD(Current Change)**

**5.<h1>Welcome to the website!!</h1>(here I have another exclamatory mark that is not present in the repository)**

**==========**

**6.<h1 style="color:blue">Welcome to the website!</h1>(here I am adding css property that is not in the local computer)**

**>>>>>>>>>>>>>>>bdfdhwhushdjsbdsd9090iids56rwtjnsansjahshagsafs(Incoming Change).**

7.<h2>Hello!</h2></body>

8.</html>

30. To remove this conflict I need both the ! and style="color: blue":

Change the line with the required modifications: <h1 style=" color: blue">Welcome to the website!!</h1> and commit the changes using git commit --am.

31. git log is useful for keeping track of all the changes that are made to our cod and keeping track of all the commits that have been made in the particular repository.

32. git reset will take the current state of the repository and revert it to the older state of the repository.

-git reset --hard <commit> meaning hard reset everything related to the commit

-git reset --hard origin/master (It takes the current version of the repository that is present in Git Hub) then I can use this to do so.

33. status, log, and reset give the information of the repository.

**Making Changes on a Git Repository:**

1. The structure is:

first, commit ->make changes->more changes-> start a new feature->keep working on the new feature. -> Fix the bug on the at some process -> We come back to fix the bug.

The solution is Branching.

**Branching:**

1. Branches are the git's way of working on different parts of the repository simultaneously.

Structure of branching:

A blue line with white dots and white text

Description automatically generated

2. The master branch is a default branch that is generally going to contain an up-to-date, stable version of the code and work on the new features we might have some feature branch we are working on. Here HEAD ->master when we try to fix the bug. The head moves to the feature when we try to add new features to our code.

3. After we are satisfied with fixing the bug and adding the new features to our code, we merge both and get the latest version of the code.

4. This is the power of the git it works on multiple files simultaneously without disturbing the other files present in the repository.

5. The **“git branch”** command tells us what branch is currently on and what branch is existing in the repository.

6. If we want to check out the new branch type **“git checkout -b name of the branch”**

For example, git checkout -b style

Output: Switched to a new branch ‘style’.

7. Again check your branch by git branch to know whether it changed the branch or not.

Example for the branching and now we are in style branch:

Code:

<html>

<head>

<title>Branching</title>

<style>h1{Color:blue;}</style></head>

<body>

<h1>This is a new Heading</h1></body></html>

8. Pass the command as git commit -am “Move Style Properties” and hit enter the code gets updated.

9. Switching the branch using git checkout master -> means that we have changed to the master branch from the style branch.

10. Code in the master branch:

<html>

<head>

<title>Branching</title>

</head>

<body>

<h1 style=” color: blue;”>This is a new Heading</h1></body></html>

11. git merge filename is useful for merging the files after performing the operations on the different branching.

For example: git merge style

The code of the file is as follows:

Code:

<html>

<head>

<title>Branching</title>

<style>h1{Color:blue;}</style></head>

<body>

<h1>This is a new Heading</h1></body></html>

**FEATURES OF GITHUB:**

Git Hub is an open-source software network.

* Forking in Git Hub Repository: Forking is nothing the copying the original repository so that we can clone, push, and pull from the repository.