**--------------------------GIT---------------------**

1. Create an empty repository

This is known an initialization

Command -> git init <repo-name>

                 git init hi-there

this command will create a new folder 'hi-there' in which .git folder will

be created which contains all the basix structure of the repo

2. create git repo to existing folder

Suppose we already have a folder name 'test'

command -> git init <specify the directory>

                 git init test/

This will create a newly initialized git repo

3. Creating a bare repository – this is used for large project where mane users push their code in the same repository

Command -> git init –bare <repo-name>.git

git init –bare big-project.git

This is basically used in the servers where many branches of the main branch are.

4. Configuring git using git config command

You need to configure 2 things in git -> username and email id

Command -> git config –global user.name “harshitjn810” -> this is the username of your github account

git config –global user.email “harsh3297@gmail.com”

git config --list will list out all the configurations

5. After git init command we have to add files to our git repo

We do this with git add command

* git add <file name> or git add . (this will add all the files in that folder)

this command will add the file to the staging area, which is a virtual area that signifies all the files to be added to the git repo

* git status -> this command will show all the files which are in staging area
* git status -s -> this command will show all the which are added and which are not added to the staging area

‘A’ will be printed infront of the file which are staged

‘??’ will be printed infront of the files which are unstaged

* git status -v -> this will get more verbose output, including what was changed in a file

You can not add empty folder to git. But there is a way to add empty folder by adding a hidden file in it.

* Git add foldername/.keep

6. after adding the files to the staging area, we are good to commit these files to the git repo

* git commit –m ‘commit messsage’ -> this will bypass the editor and commit with specific message
* git rm --cached <file name> -> this command will delete the specific file from the fit repo
* git commit –a –m ‘commit message’ -> suppose after staging a file, you have done some changes in the file, so in order to commit all those changes you will use this command

7. Tag Command

-> git tag –a <tag name> -m <message>

git tag -a v0.1 -m “this is the annotation tag”

* command -> git tag 🡪 this command will list out all the tags within the repo

8. Branch

-> To create a new branch

Command -> git branch <branch name>

git branch testbranch

* To check how many branches we have other than master branch

Command -> git log --oneline –decorate

* To change from one branch to another branch

Command -> git checkout <branch-name>

git checkout testbranch

* To delete a branch

Command -> git branch –d <branch-name>

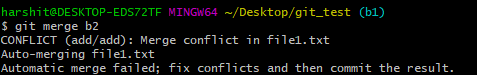
* To merge a branch -> go to the branch where you want to merge the files. And then run the following command

Command -> git merge <branch name from where you want to merge the data>

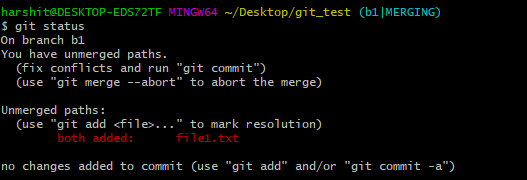
**Merge Conflict**

Suppose you have the same file name in both the branches, i.e from where you are merging and to where you want to merge.

In that case we will face merge conflict.

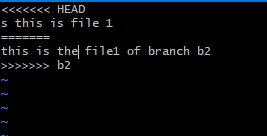


We can check what went wrong with git status command



Here the data of both the files are present in file1.txt

Open this in vim



This is the content of the file.

Head is pointing to the branch you want to merge the data and there is a separation with ===== between the data of 2 files

Just edit the file, i.e how you to merge the 2 data and then save the file, run git add command and then git commit to commit the file in the branch

* To check the log file

Command -> git log

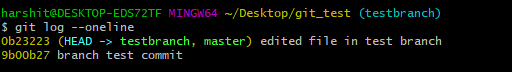
* To revert the commit

Command -> git revert HEAD

This will commit the reverse of the out commit

To see what all commits happened on that particular branch

git log –oneline



With the commit id

So suppose if you want to delete the 4th commit from the head commit

Command -> git revert HEAD~4

* To clone a local repository

Command -> git clone <local repo> <new repo>

Git clone testbranch newbranch