**GIT**

1. Git is the local version control system
2. We can make to transfer data to the server by the help of the websites like github
3. The git bash is the default terminal for the git and it is based on the linux terminal but we can also use the other terminals also
4. In git bash the command **pwd** is used to goto the present working directory
5. In git bash the command **cd** is used to change the directory, it is same as in cmd
6. In git bash the command **ls** is used to get the list of the file in that folder
7. First we have to install the git
8. Then we have to open an folder, where we want it to make its reprository
9. Then we have to configure git for that folder by right-clicking we get the option to open it with git bash and in it we have to write the following statements :

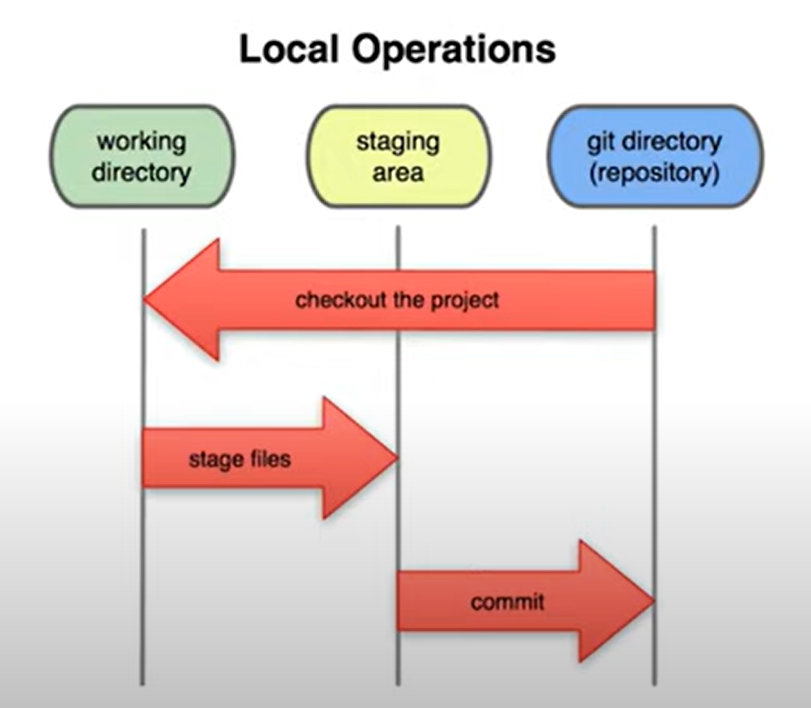
git config --global user.name “neel”

git config --global user.email [neeldevenshah@gmail.com](mailto:neeldevenshah@gmail.com)

1. To check the repositories present in the computer write the following code in the command prompt or git bash :

**git status** (Very Important, it will give status of git)

1. The git has the three stage architecture :



🡪In staging area we can decide which file do we have to sent to the git repository from the working directory, Hence we have the option for making some or all file go to git repository

1. We can also delete the .git repositories by going to the folder where that exist and there following steps to be followed: (There also another method defined with help of code check it)
2. Click on the three dots on the navbar
3. Click on options
4. Click on view on navbar
5. On list go to the folder-shaped icon
6. And in that click on the show hidden files and folder
7. And there the folder named .git will be visible
8. Delete it and hence the repository will be deleted
9. Then change the settings we changed to the as it was(Hidden files are not seen)
10. We can make the new git repository by the following code in the cmd or the git bash :

**Git init**

🡪But make sure that you open the git bash by clicking(right click and select open in git bash) it from the folder you want to make the repository

🡪We can also use the git GUI application for doing this type of things

🡪And after the making of the repository the address will come make sure that address is same otherwise delete it and make new one

🡪For neel’s laptop make sure that the git repositories are made at : c:/Users/Hello/git repositories

🡪The git repository is made by the name .git

1. Now after adding the repository to the folder, by checking the git status in cmd or git bash we get the list of the files in the folder as the untracked files
2. Now as we know the git has three stage architecture, And by making the files in the folder we make the first step completed
3. Now to add these files to the staging area we have to write the code in cmd: **git add --a**

**🡪**To check the files we write the git status and we get the list of the files with the title the changes to be committed.

**🡪**If not then check the command written before

**🡪**To make the files to be upstaged we make the following code :

git rm --cached <file>...

1. Now as we know the third stage is to add the files to the repository and this is done by the help of the **commit**

**🡪**We can make the commit by the help of the following statement :

git commit -m “Initial Commit”

🡪And we can check if the commit is taken place or not by help of : git status

1. We can also get the details of the commit by the help of the: git log
2. If we make the change in the any files than it will show up while we write the git status
3. In which if we have two options to be done :
   1. git add <file>… to update what will be committed
   2. git restore <file>… to discard changes in working directory

🡪If we want to work with the all the files or more than one file than we can use **git add --a** and due to it the mass change will take place

🡪**And after doing the change we have to make the commit done to make it final to the repository**

1. (I.M.P.)We can also remove the depository from the machine by opening that folder and then using the code in the git bash: **rm -rf .git**

🡪Caution the all the data stored in the repository will be deleted

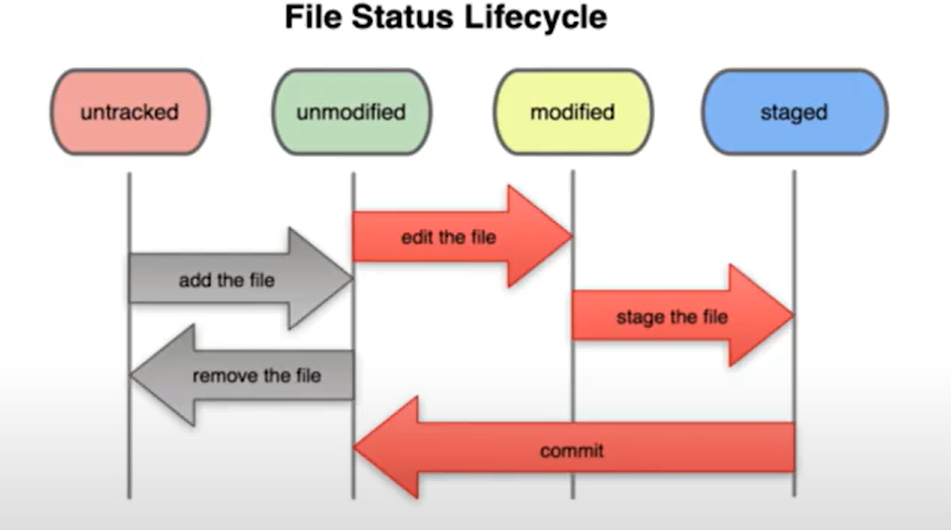
1. The meaning of the clone is the copy in the object oriented programming
2. We can also get an project from the remote repository(i.e. github) to the local repository(i.e. your machine)

🡪The steps for this process are :

1. Go to the github.com and select an project which you want to import to the local repository
2. Click on the button available on the upper right hand side with written clone or download and having color green and then copy the link given in that
3. And then go to the folder where you want to copy that project and there open the git bash and write the following code :]

git clone <paste-link-of-project-here>

1. And then wait till the repository is downloaded
2. File Status Lifecycle



1. We can ignore some file or some folder while committing the project to the repositories, i.e. It is not included in the repository and space is saved

🡪Usually the file that are ignored are log or some data file and etc

🡪The steps for doing the ignore are :

1. Open the git bash from the folder where you have your project
2. Now for making the .gitignore file we have to write the following code :

touch .gitignore

1. And the file will be maded in the project folder
2. Now open the .gitignore file and in it write the name of the file or the folder which you want to ignore(for folder /dir or etc)
3. And the file/folder will be ignored

🡪We can also mass ignore the any type of the file by putting the following code in the .gitignore file : Eg. **\*.log, \*.txt, etc**

1. If any file is in the staging area and if we again do the change in the file than the file is been showed in the both the areas staging and modified

🡪And when we add the file to the commit then the file which is at the staging area will be committed but the file which is in the modified area will be remained untouched

1. By the help of the **git diff** we can compare the files which are in the stagging area and the modified area
2. By the help of the **git diff--** we can compare the files which are in the stagging area with the files which are committed before
3. If we do not want to follow the change in the any file then we can use the code like :

**git rm --cached <file-name>**

1. By using the **git log -p** , it will show us what things are being removed from the file in detail for each commit
2. By writing this code **git log -p -3** it will show the last 3 commits that has been made in detail

🡪Here the number which is 3 can be changed by any number

1. By the usage of the **git log --stat** it will show the details of the all the commits in the short
2. By using the following code **git log --pretty =oneline** we can get the name of the commit and details of the commit in the one line
3. By writing the code **git log --pretty =short** we can get the details of the commit and its maker in short, Here date and other details will not come
4. We can also get the data of the commits that has been done in the specific time by the help of the code : git log --since =2.days

🡪Here the days can be replaced by the months and years as per requirement and we can also change the digit that is used

1. We can also use the statements written by us like: **git log --pretty=format:”%h -- %ae”**

🡪Here the **%h, %ae** are the different shortcuts for different things by the help of which we can get an filtered result

🡪We can get different placeholders like this and their meaning by :

1. By searching the **git scm useful options for git log format** in the google
2. And then in that with help of the find function in the website find the following sentence and will get the list of the placeholders :

**Placeholders that expand to information extracted from the commit:**

1. To remove an file from the staged stage write the code like : **git restore --staged <file>**
2. If we by mistake make the mistake in the file which is on the live project, Then by the help of the following code we can make the old committed file back in the project :

**git checkout --<file-name>**

**🡪**By the help of the following function the old file will be deleted and the new will be downloaded

**🡪**But Be cautious the all new code written will be removed

1. For removing all the changes you made in the project and if you want to import the old committed project than use : **git checkout -f**

**🡪**But Be cautious the all new code written will be removed

1. **Pull** = **Get** the code from the github to the local computer
2. **Push** = **Give** the code to the github from local computer
3. In git bash we can paste an code by the help of the **shift + insert** and **copy an code by simply selecting it**
4. We can make the code to push to the github by making this three code by using the git bash where the project exists :

git remote add origin https://github.com/NeelDevenShah/Box2.git

git branch -M <branch-name>

git push -u origin <brach-name>

\*Here instead of the box2 which is an repository name, we have to add the **repository name in which we want to push**

1. If we pull or push the projects than the commit’s related to the projects remain intact
2. We can also make the short cut keywords in the git by the help of the alias example :

git config --global alias.ci commit

🡪Here from now we can use the git ci instead of the git commit to commit

🡪In this you have to define unique keyword for each of command and put instead of ci and the thing for which the command is to put it instead of commit

1. In git we can make the new brach by the help of the following code :

**git checkout -b develop**

🡪Here the -b indicates that the brach will be seprated

🡪The develop indicates the name of the new branch and the name of the branch can be anything

🡪We can also make the seprate branch from the master brach and also can combine the new brach with the master branch

1. We can also work with more than one branch at the one time

🡪Example we have two branches master and develop

🡪After making the new brach we are now transferred to the develop branch

🡪Now after saving it we can switch to the another branch by the help of the code like :

**git checkout master** (in git bash)

🡪And now we are switched to the other branch and we can work on it

🡪In this example we have to write the branch name where we want to go

🡪There can be many number of branches and one can work on many number of branches

1. We can get the list of branch present in project by writing : **git branch** (in git bash)
2. We can also merge the two branches at a time and make the combined branch which is would be known as the master branch
3. The code for the merger of the master branch with the other branch is as follow:

**git merge develop**

🡪Here the develop used can be be any branch name that you want to merge

1. While merging the two branches some time the conflict also arises and that conflict can be resolved by going into the editor (In this case vs studio)(You can easily perform this type of function on the github)

🡪And when we open we will get the lines with the two different colors the one is for the current change and the other is for the incoming change

🡪And above them there is the options like accept current change, accept incoming change, accept both, etc

🡪By choosing one of them the merger error can be solved

🡪And then write **git status** in the git bash

🡪then it will give that the file is modified and then follow the steps to commit it

🡪And then finally the brach will be merged

1. We can get the list of the branches that are merged by the following code:

**git branch --merged**

1. We can get the list of the branches that are not merged by the following code :

**git branch --no-merged**

1. We can also delete an existing branch by the two ways
   1. The error will be showed(and cannot be deleted before marging)

**git branch -d issue2**

* 1. The error will be not showed

**Git branch -D issue2**

\*Here the issue2 is the name of the branch and it can be changed to the name of the any other branch

1. In git we can also make the merge the branches other than the master branch and make new master branch and old one will be deleted
2. In any project mainly there are mainly three type of branches:
   1. Master
   2. Development
   3. Topic branch
3. We can also find the remote repositories linked with the local git repository by the help of the following code : **git remote -v**
4. (I.M.P.)While pushing an project from git to github all the branches are not pushed only the one branch is pushed which is opened by the code: **git checkout master**

**git push origin master**

\*Here the master is the brach name

🡪And the only master will be pushed to the github not all the branch and we have to change to first change to branch which we want to send, before sending

1. By the help of the git bash we can also delete the branches in the github by the help of the following code : **git push -d origin <branch-name>**