**GIT AND GIT HUB**

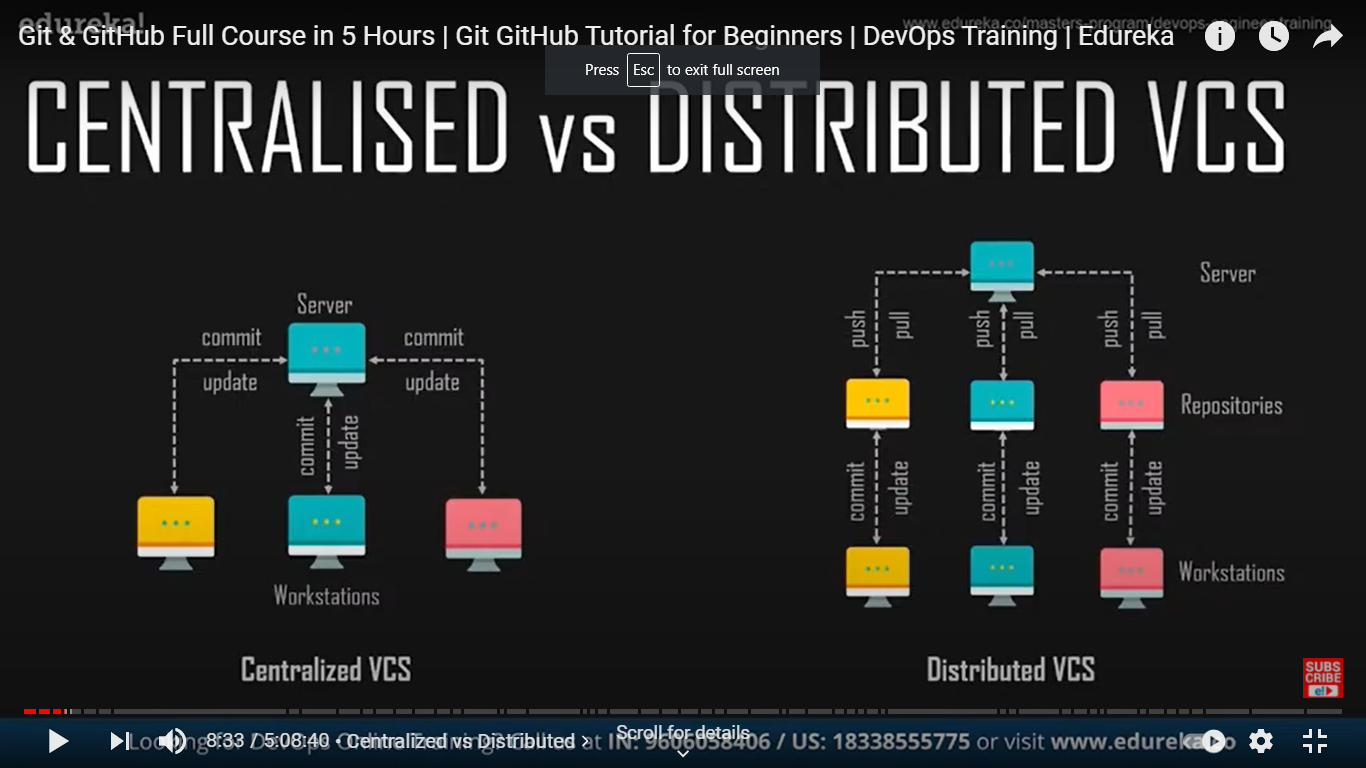
git🡪 version ctrl sys or source code ctrl system

version ctrl sys 🡪 three type local, centralized and distributed

local🡪 stores info about the repositories locally on a machine

centralized🡪 stores all info in a centralized platform

repositores 🡪 we can create repository for a folder or director then store everything in the rep, the files can be anything like image audio video files

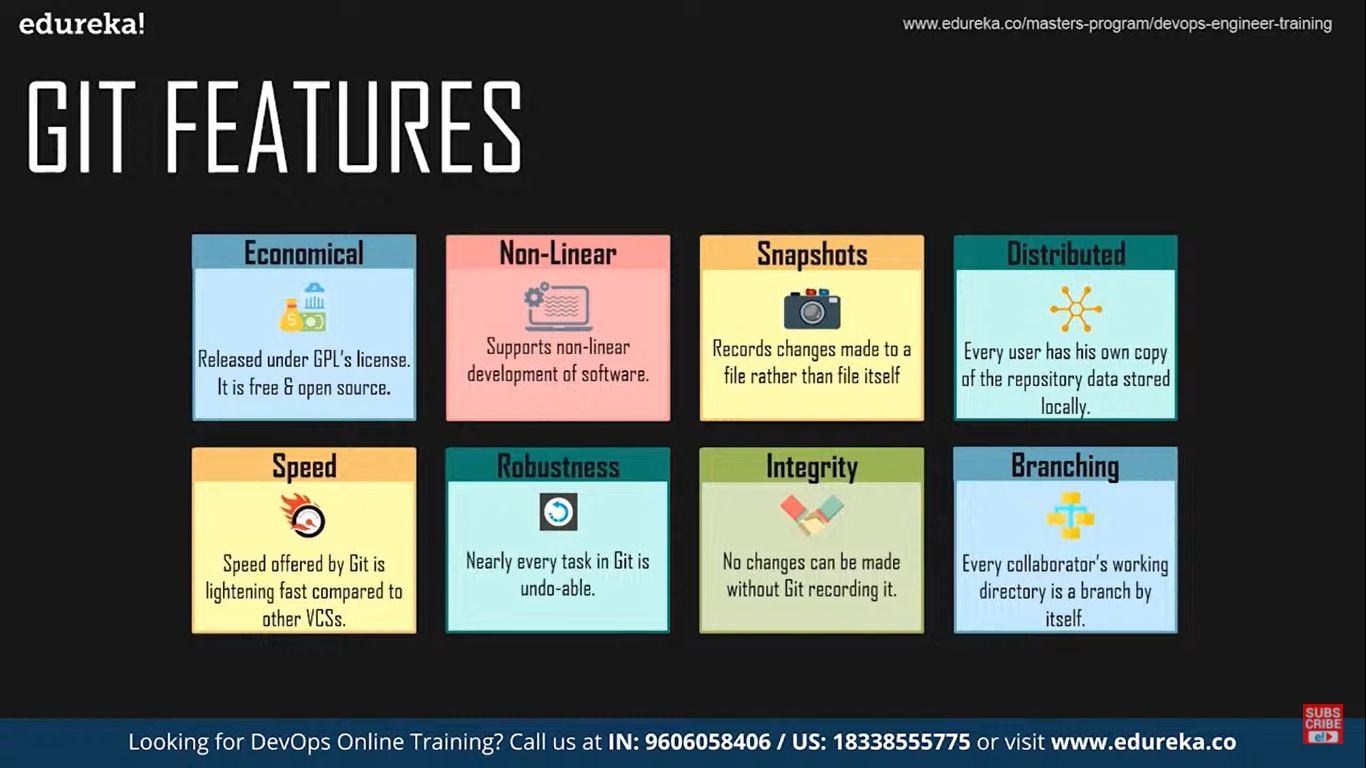


central server 🡪 a src code is managed there, then diff computers remotely connect to the repositors

a user should connect to the server update the code and commt the changes

distributed🡪 each user has a local repo once doing all updation the file can be sent to the central server

GIT🡪 distributed version ctrl sys, u take a repo work on it then share it to get, where others can use your code or file



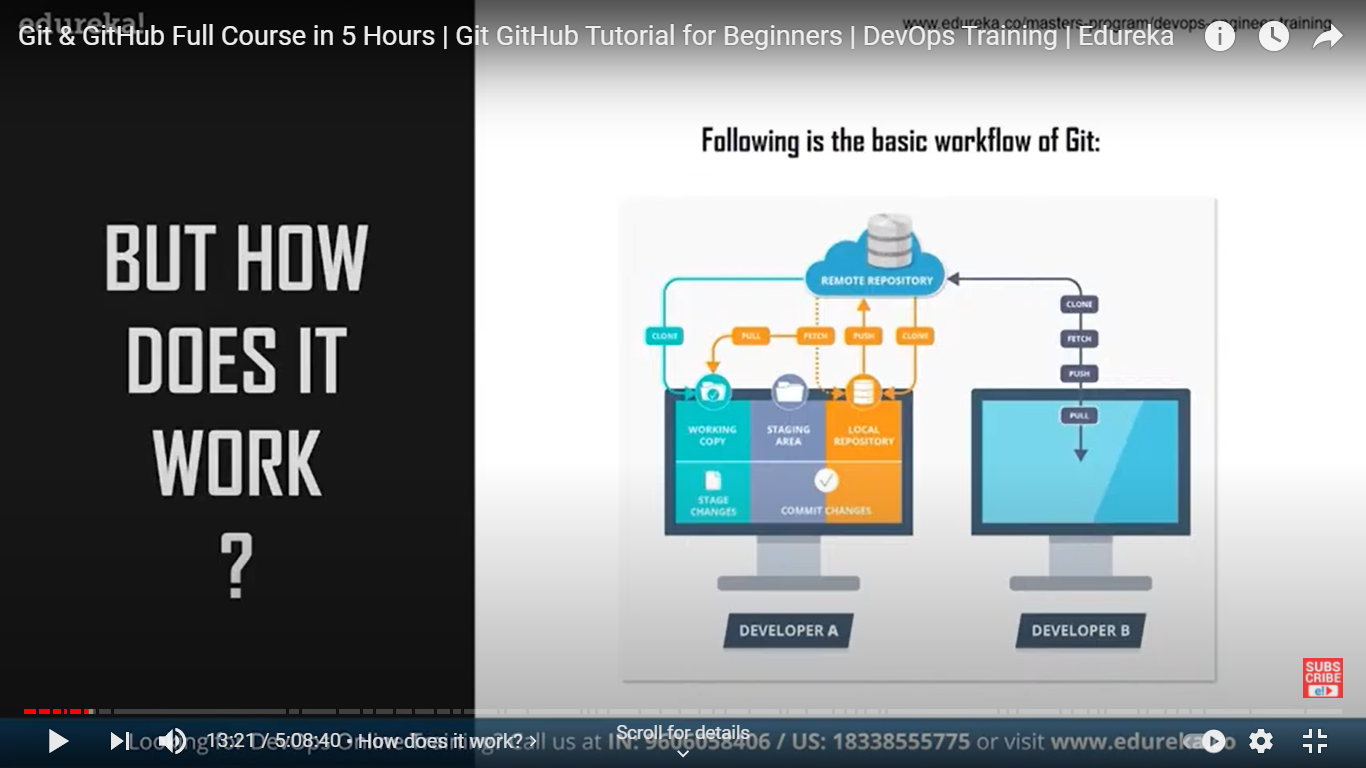
local repo🡪 in you computer

remote repo🡪 available for every one

first🡪 work in a file

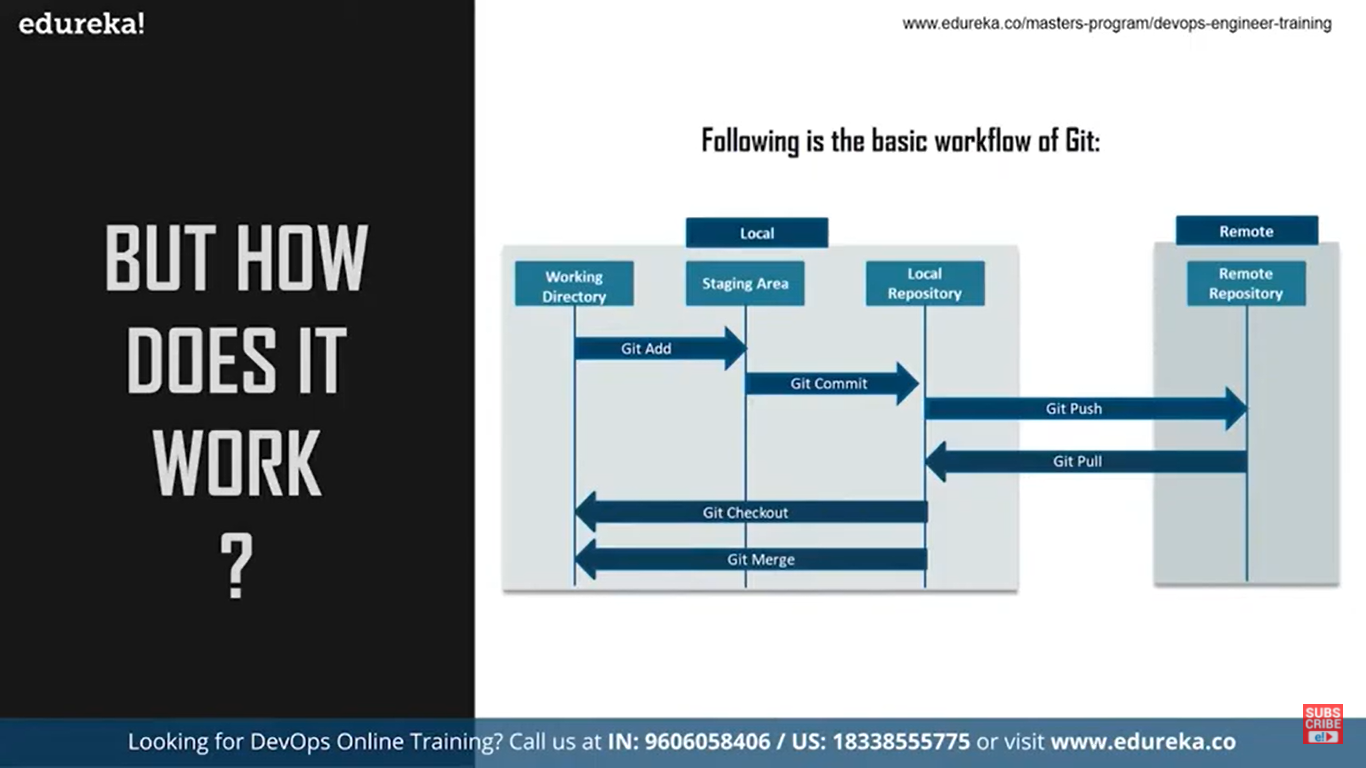
save the changes then commit the changes so that all the changes will be stored in local repository

then clone or push the local repository to the remote repository, we can also clone or pull data from the remote repo to local repo

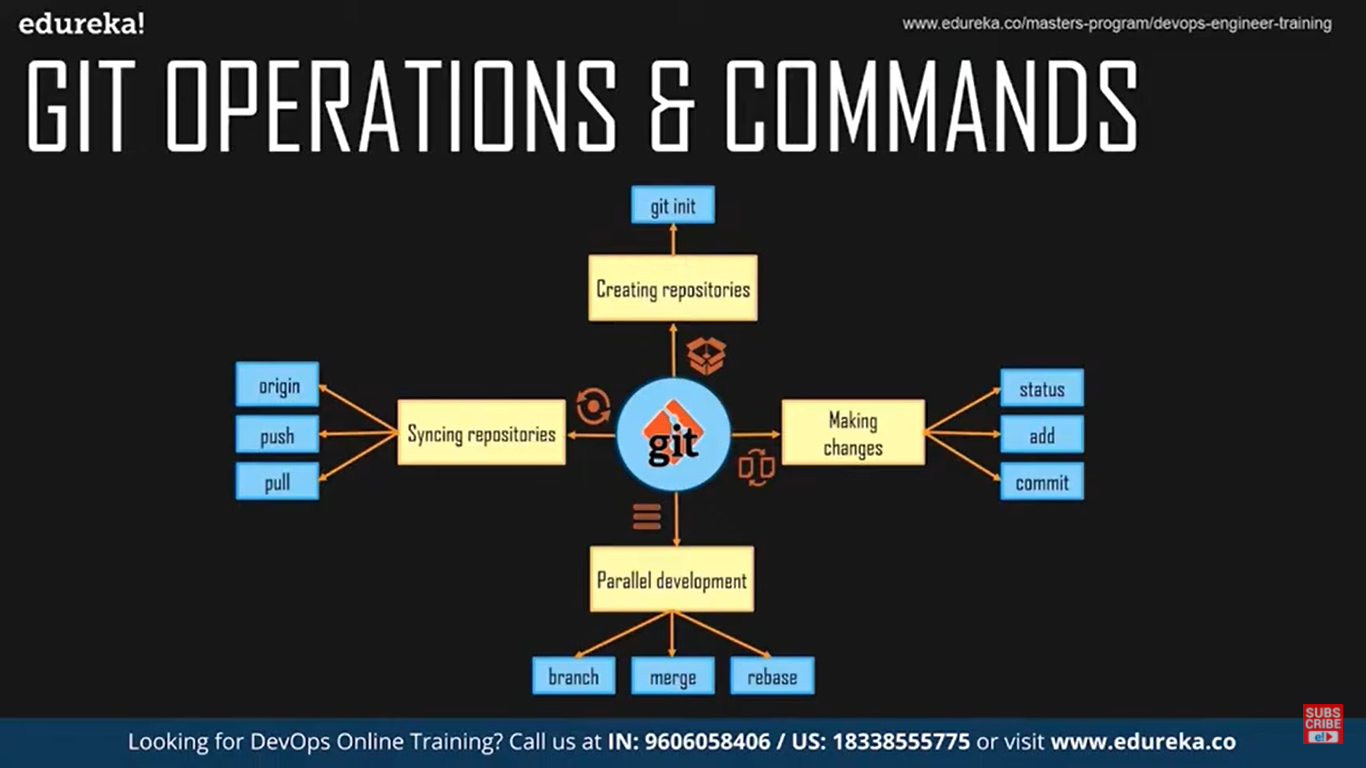


basic work flow

only when we commit the changes are pushed to local repo, if not t is present in a temporary area (memory) called staging area, after pulling any file from remo repo we can merge them with what files we have

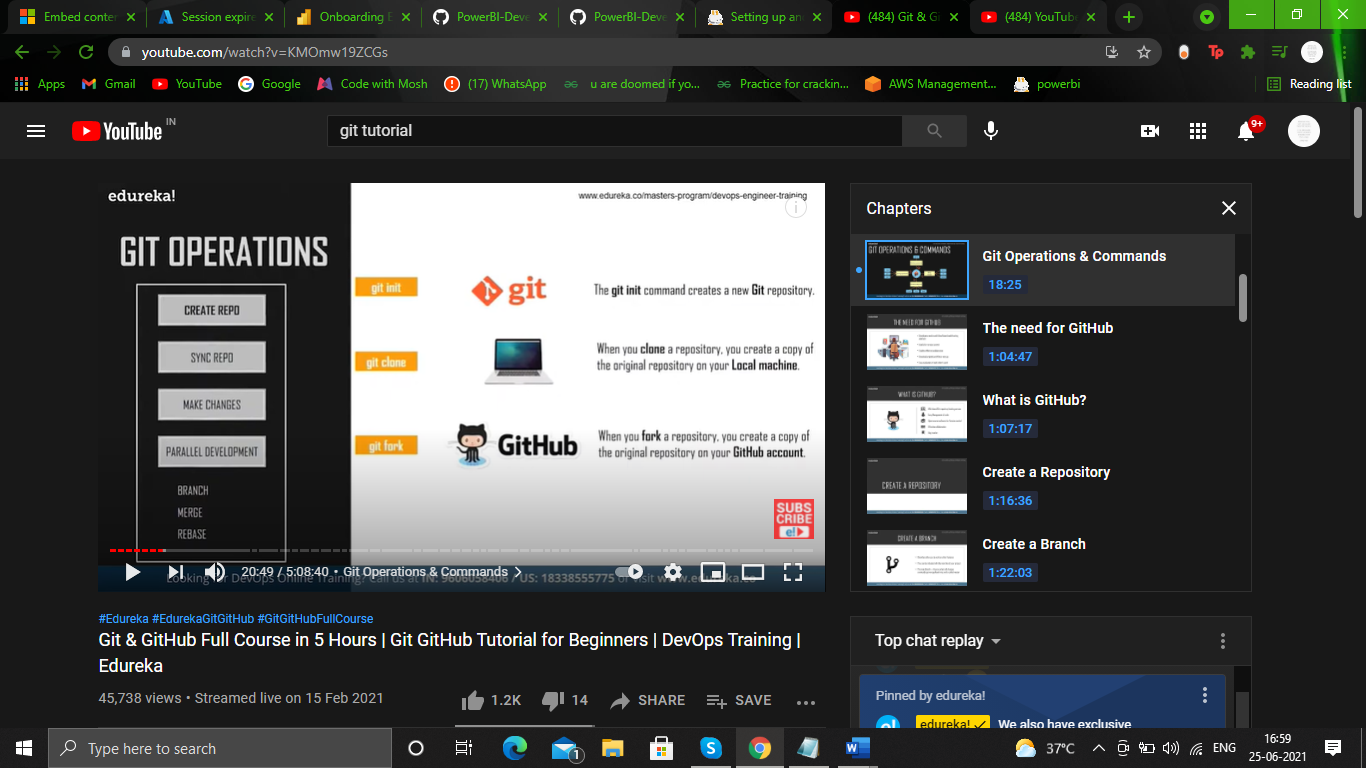
\

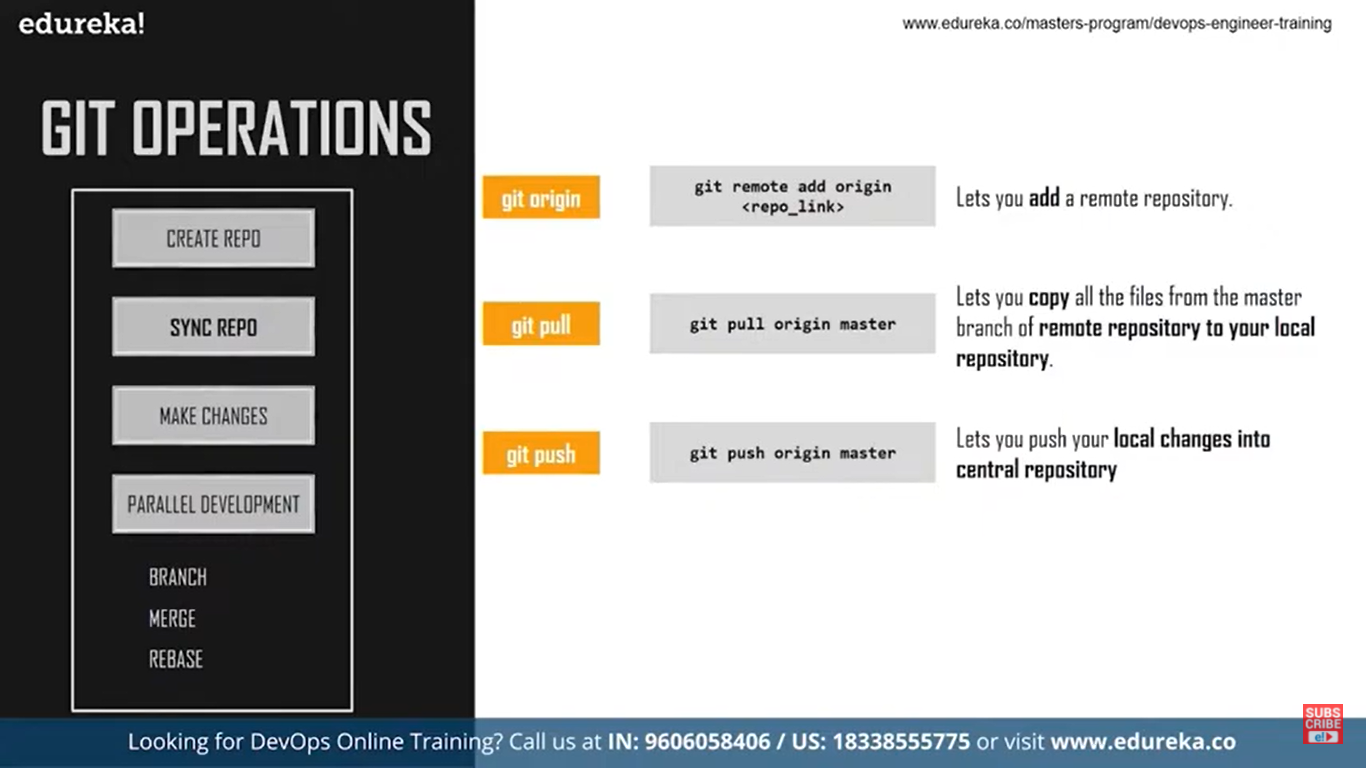
COMMANDS



**init🡪 to create repository in your pc (local)**

**branch🡪 to work parallely with multiple workers**

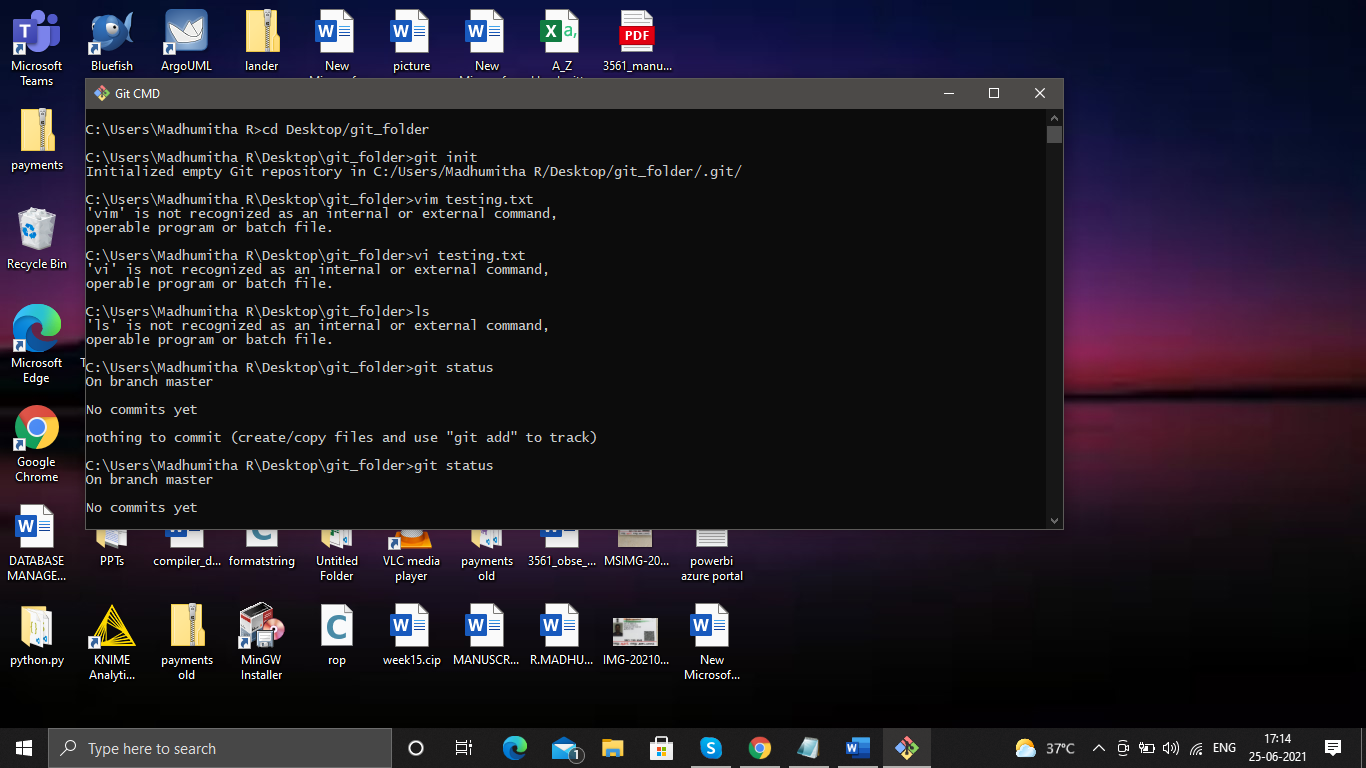




origin🡪 to add a remote repo to our local repo

to configure a git repo, so that any changes made will be reflected we can configure the global variables like name and email

* git init
* git config –global user.name “somename”
* git config –global user.email “email address”

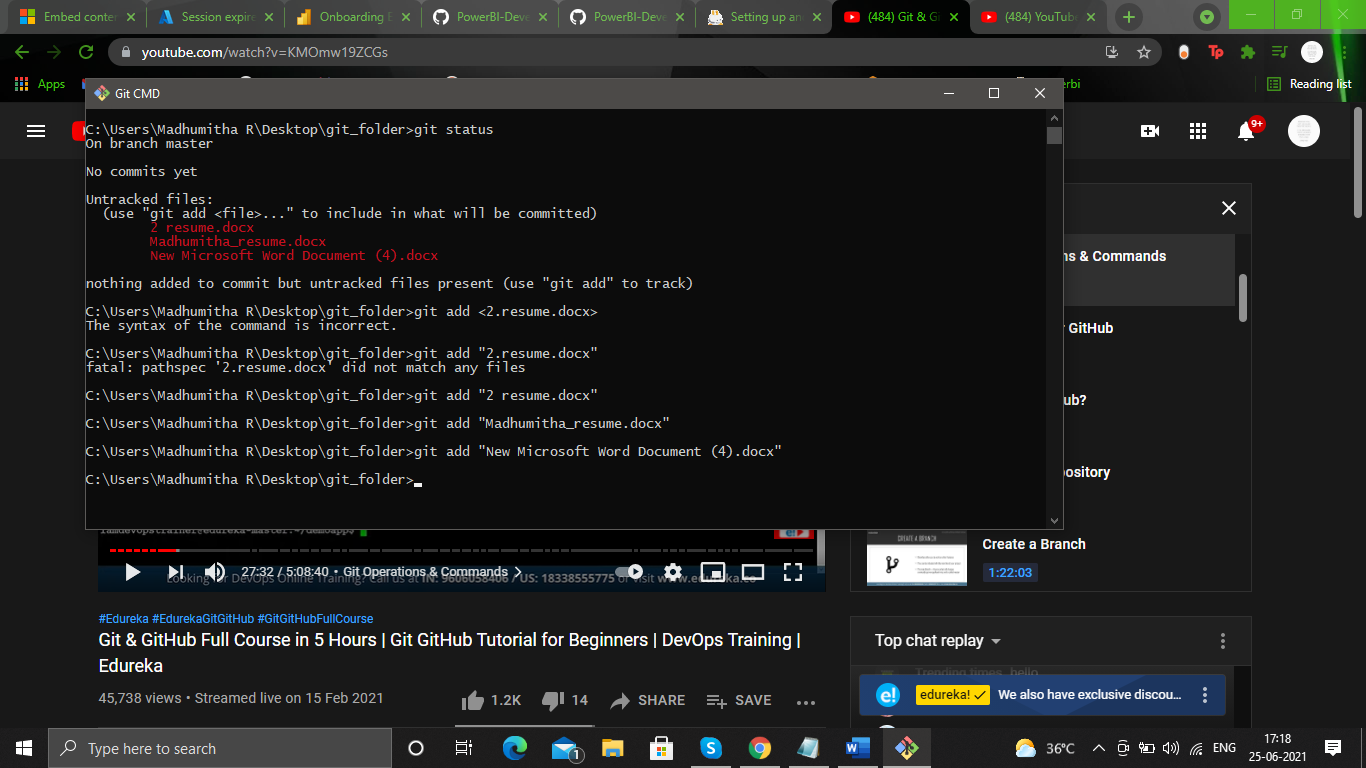


open git cmd 🡪 then go to the directory you want to create a repo

1. **give the git init comand**
2. a git folder will be created in the folded
3. then add files or anything in the folder
4. the git will keep track of it, but first we need to make sure that all the files we added newly must be moved to stage area to do so give
5. **git add “filename with extension”**
6. **git add file1.txt file2.txt fil3.py** 🡪 multiple files separated by a space
7. **git add .** 🡪 to track all the files
8. **git add ‘\*.txt’** 🡪 all files with extension txt

every time we add a new file, the git doesn’t track it so first put the file in the staging area to be tracked by git

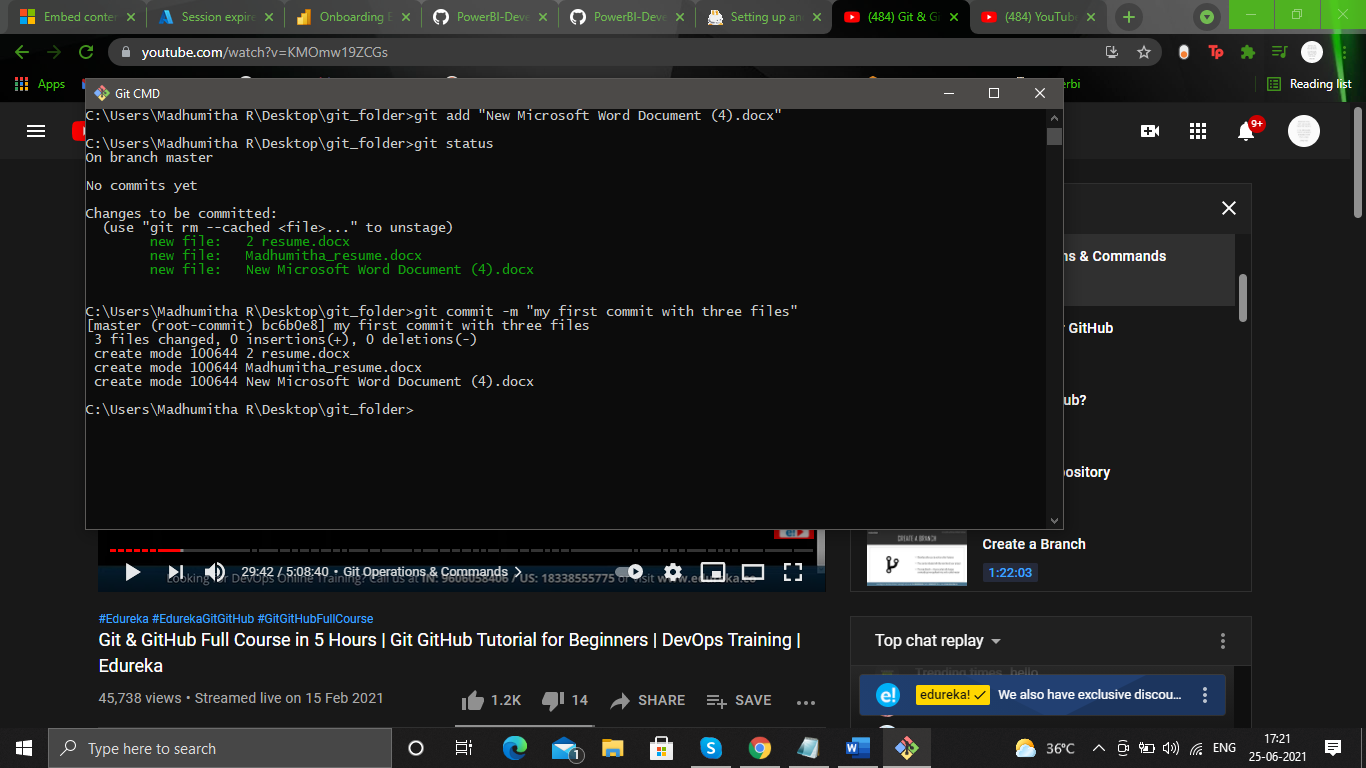
**now the files will be tracked git status will give the list of the files that are untracked**



**now all files will be tracked**

as soon as we commit the file, in the stage area the file doesn’t vanish, it stays there until the file is changed or deleted, the atage area is a temporary storage area so we save the files in that area before committing so that we can review once before we commit,while working if we find that we need to bring back what we did by mistake we can take it from stage area,

**COMMIT**



**git commit -m ”some name for the commit”**

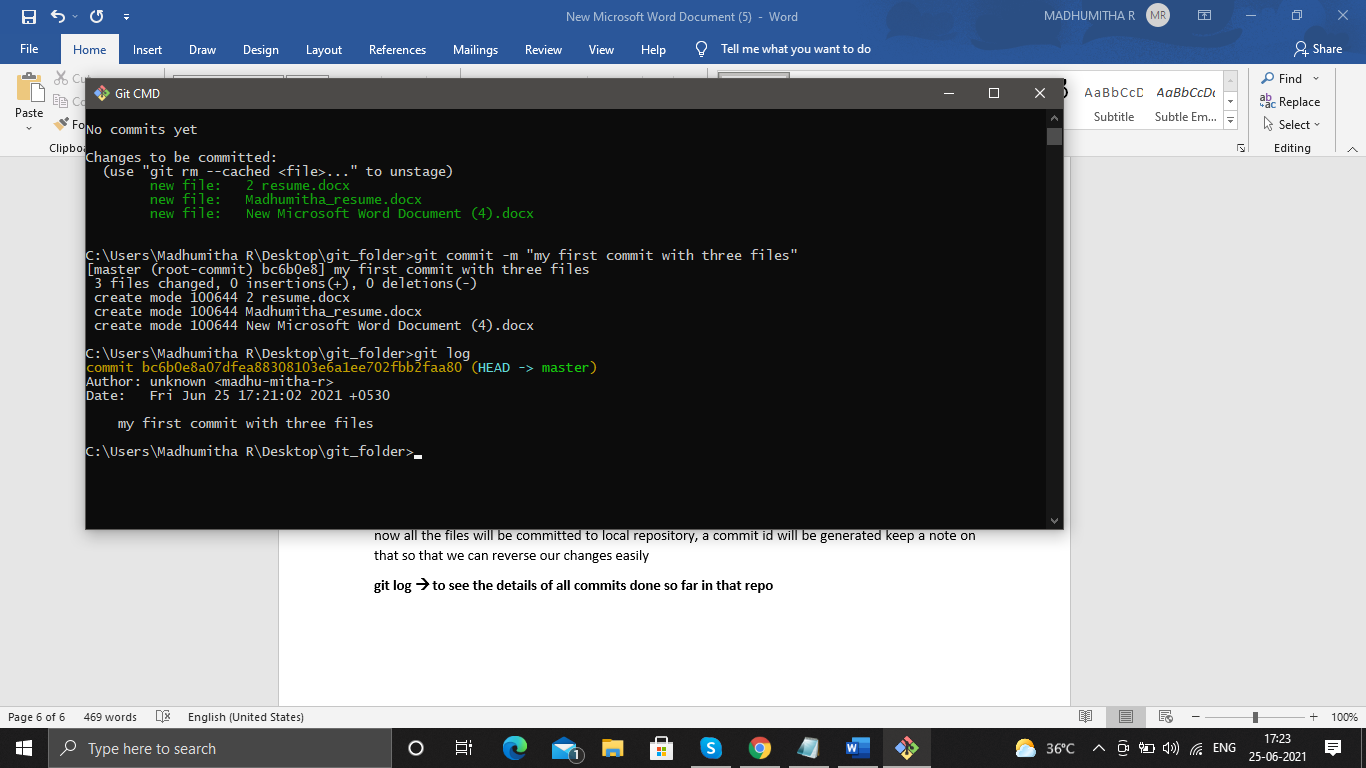
**-**m 🡪 message

now all the files will be committed to local repository, a commit id will be generated keep a note on that so that we can reverse our changes easily

**git commit -am ”some name for the commit” 🡪 to commit all modified files**

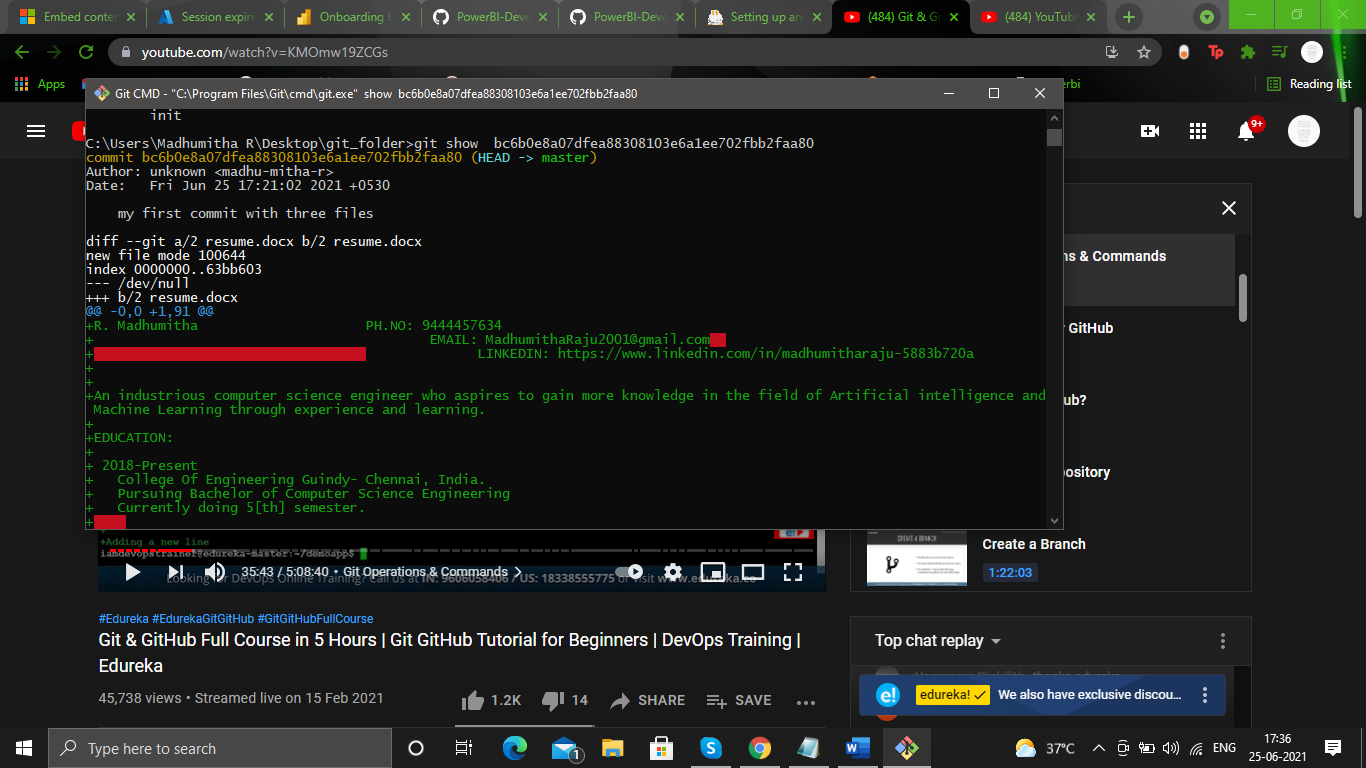
**git commit🡪 will open a text editor we can give a description about the commit there and save the changes too**

**git log 🡪 to see the details of all commits done so far in that repo**



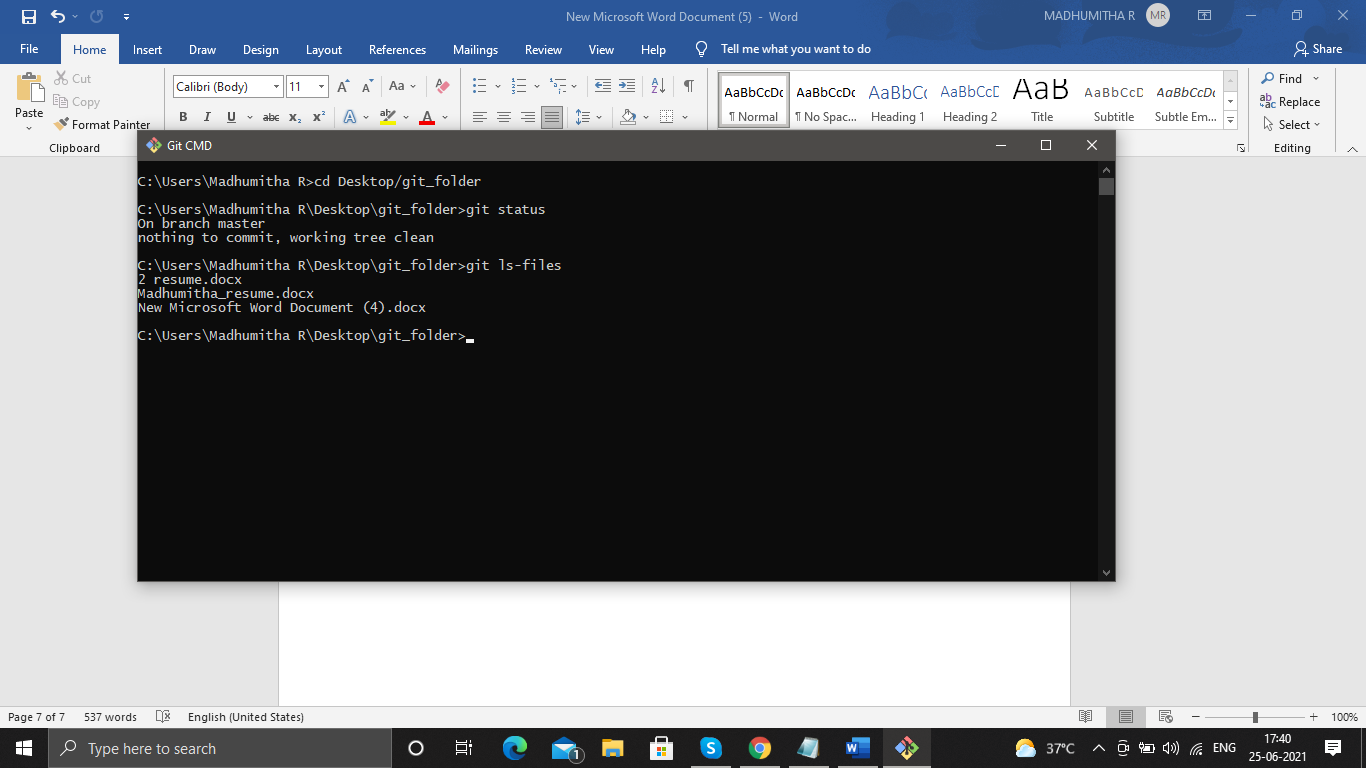
commit id the author and the name of the commit will be displayed

git show commit id 🡪 will give further info about the commit like all the places where the code or the content has been changed , but if a new doc is include then the entire content of the doc is displayed



if we give enter enter enter….. then all the content will be displayed

* **git ls-files🡪 list of all files that are being tracked**



remember we can also commit a file before staging it but its not a good practice, we must always need a staging area

* **git remote add origin link of the remote repo**

will add the remote repo to the local repo that we are currently working on

for the remote repo create a repo in github and copy the link here in the command

**origin**🡪 alias name of the repo in the link,

so we add the repo in the link and the repo we are currently working at, using git

* **git remote -v**

to check if the repos are linked, we use this cmd

* **git push origin master**

so we are pushing the repo we are working to the remote repo using the aias name origin for the remote repo , we are actually pushing the master branch, to the repo, a master is a branch that created by default while we create the local repo

**REMOVING FILES**

delete it manually, so that the deleted file is in the staging area and not in the workstation so give

**git add deleted file name**

this will remove the file from the staging area and not in the local repository

to remove the file from loc repo commit the changes

**git commit -m ”deleting unused files”**

**another method:**

**git rm file1.txt file2.txt**🡪 removes the files from the staging area as well as from the loc repo no need to commit the changes

**TO RENAME OT MOVE FILES**

**mv file.txt main.txt 🡪 normal linux command**

file.txt is deleted and main.txt is added both changes should be committed and staged so

to reflect changes in stage area

**git add file.txt**

**git add main.txt**

to commit

**git commit…**

OR method two USE

**git mv file.txt main.txt**

**git commit…**