## GIT installation:

Download git from git link

Run the file, follow the easy steps

$ git --version

Will give you the current version of the git installed in the machine

Create a new project in the desired drive

Then

$ navigate to the project folder

Now add files in the local workspace/folder

Ex: a.txt and b.txt

Now initiate the repository

### $git init

Initialized empty Git repository in Drive/project/.git/

It is a hidden file(you can view them by viewing hidden files option

### $git add

This command will add the files to staging location

### $git status

This will give you details that what files are tracked and what are not tracked

Before committing the changes for the first time, user has to identify by giving email id and name

$git config --global user.email “emailed”

$git config –global user.name “username”

$ git commit -m “provide comments”

Now the files will be committed to local repository

Now check status

$git status

On branch “branchname”

Nothing to commit, working tree clean

### $git log

This will give you info on how many commits are done and who did them at what time

Once the files are added and you modify them

Instead of adding and committing them, you can use the below command which adds the files to staging and committing

Git commit -a -m “message”

### $git diff

This command is used to find the difference between the files at different stages

$git diff “filename”

This compares file in working directory and staging directory

$git diff HEAD “filename”

This compares file in working directory and last commit in the local repo

$git diff --staged HEAD filename

Or

$git diff --cached HEAD filename

This compares file in staging to last commit in local repo

$git diff commitid filename

($git log ------ this gives commit id)

This compares file in working directory to specific commit in the local repo

$git diff –staged commitid

This compares the file in staging directory with specific commit in the local repo

$git diff commitid1 commitid2 filename

This compares the files in specific commits in the local repo

$git diff master test

This compares the files in two brances

$git diff master origin/master

This compares differences in local and remote repo’s

### $git rm

This is used to remove files from wd and staging

### $git ls

Will give the files in wd

### $git ls-files

Will give the files in staging

$git rm file1.txt

This will remove file1.txt from wd and staging

$git rm -r .

This will remove all the files from wd and staging

$git rm --cached file1

This will remove file1 from staging only(not in wd)

$rm file1.txt

This will remove file1 from wd only

### Checkout command

We can use checkout command to discard unstaged changes in the tracked files of wd.

$git checkout – file1.txt

This command discards the changes in the file file1.txt

### Reset command

**To remove changes from staging area**

When the file is added to staging area using add command, you can revoke it from staging using

$git reset filename.txt

Now the file will be removed from the staging area

**Undo commits at local repo level**

#### $git reset --mixed commitid

(for example commits are like c1,c2,c3,c4. Now you want to remove the commit c4 then you need to pass the commit id of c3)

This removes the changes in the commitid mentioned in local repo and staging area

$git ls-files for checking files in staging. $git log --online for checking commitid and its comments.

#### $git reset --soft commitid

This removes corresponding files in local repo only.

#### $git reset --hard commitid

This removes corresponding files in local repo, staging and wd.

## Git branching

To view the branches

$git branch or $git status

To add a branch

$git branch branchname

To switch to specific branch

$git checkout branchname

To create a new branch and switch to that branch

$git checkout -b branchname

### Example

$git touch a.txt

$git add a.txt: git commit -m “commit file a.txt”

Now in master branch there is a file a.txt

Create a branch now  
$git checkout -b br1 (new branch br1 is created and switched to br1)

Now check the status of files and commits

$ls (you will find the file a.txt, which is created before creating the branch)

$git log --oneline ( you will see the commits made in master branch before creating the branch br1)

Add files in branch

$git touch b.txt c.txt

$git add b.txt: git commit -m “commit file b.txt”

$git add c.txt: git commit -m “commit file c.txt”

Now switch to master and check the files in it

$git checkout master

$ls

Observe that a.txt only present in master branch as the new files b and c are added in only br1 branch.

## Git Merging

### Fast forward merging

No changes happened in the master branch after creating a branch, then merging the child/feature branch to master is termed as Fast forward merging, there won’t be any conflicts in this mode.

#### Example

$git init

$touch a.txt b.txt

$git add a.txt; git commit -m “master1commit”

$git add b.txt; git commit -m “master2commit”

$git branch -b featurebranch

Now user is switched to feature branch

$git log --oneline

You can see the two commits made in master branch

$ls

You can see the files a.txt and b.txt

$touch x.txt y.txt

$git add x.txt; git commit -m “feature1commit”

$git add y.txt; git commit -m “feature2commit”

Now merge the code of feature branch into master, for this you need to be in master branch

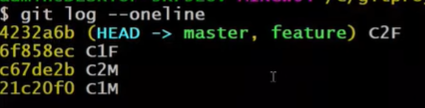
$git checkout master

$git merge featurebranch

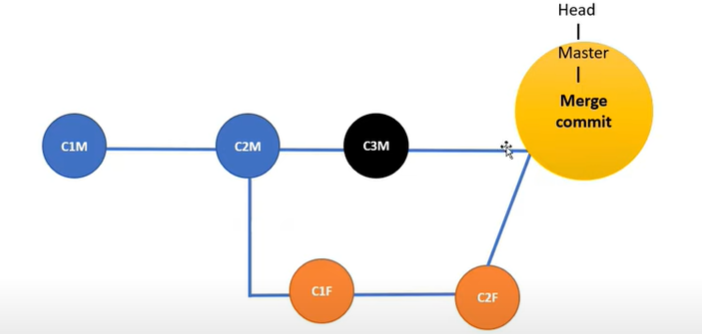
Now check the number of commits made

$git log --oneline

You can find there will two commits (2 from master and 2 from feature)



### Three-way commit



#### Example

$git init

$touch a.txt b.txt

$git add a.txt; git commit -m “C1M”

$git add b.txt; git commit -m “C2M”

$git branch -b feature

Now user is switched to feature branch

$git log --oneline

You can see the two commits made in master branch

$ls

You can see the files a.txt and b.txt

$touch x.txt y.txt

$git add x.txt; git commit -m “C1F”

$git add y.txt; git commit -m “C2F”

Now switch to master branch and create a new commit with a file

$git checkout master

$touch c.txt

$git add c.txt; git commit -m “C3M”

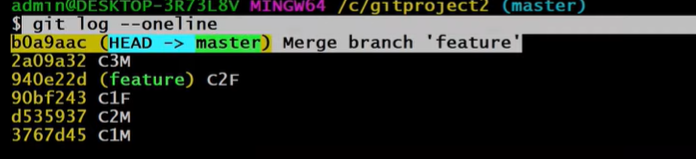
Now merge the feature file

$git merge feature

Vi or default editor will be opened, type wq! And press enter

Commit will be done

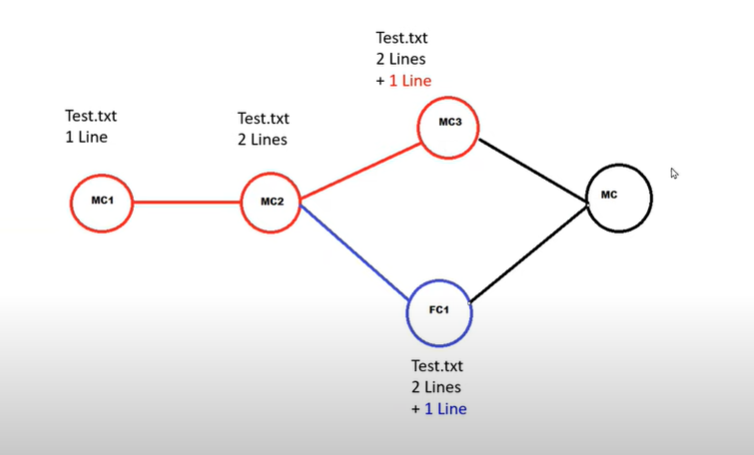
Now total commits will be 5

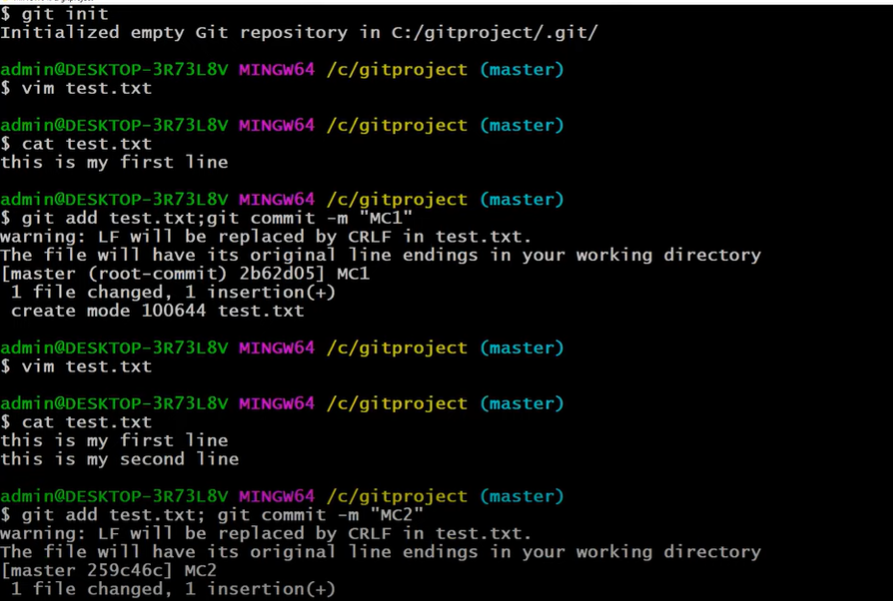


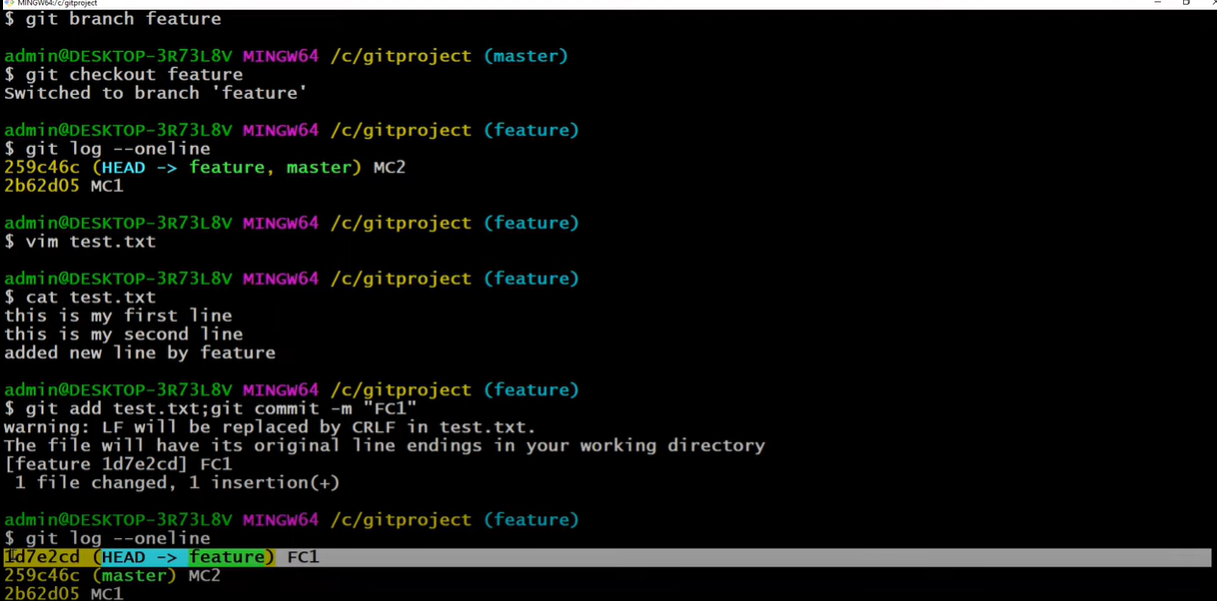
Master branch Head will be the new merge commit

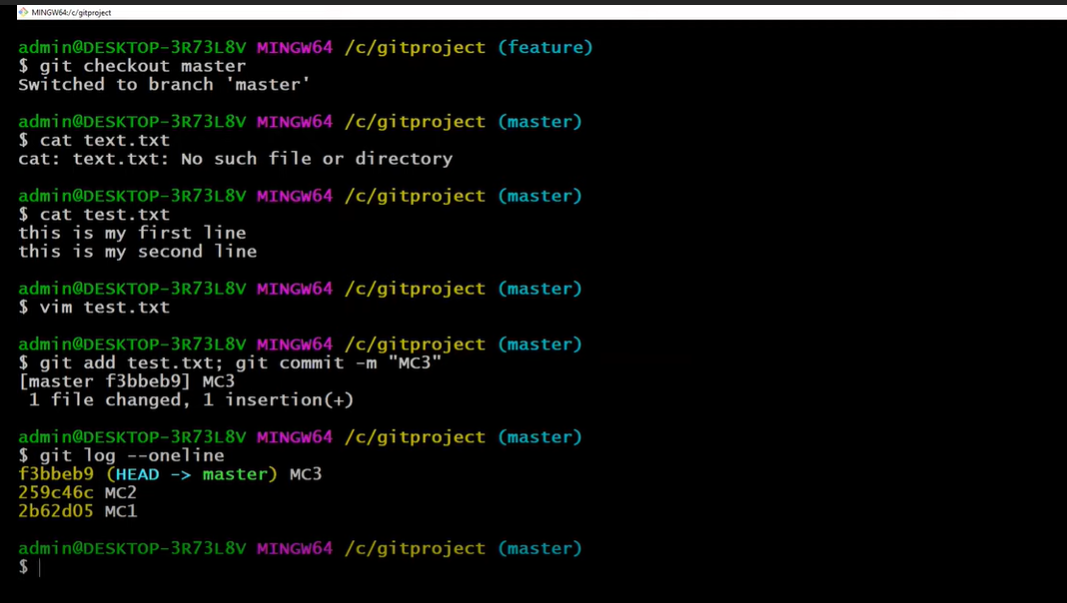
For feature branch the Head will be second commit made in that branch

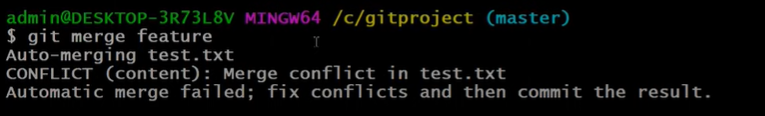
### Merge Conflicts



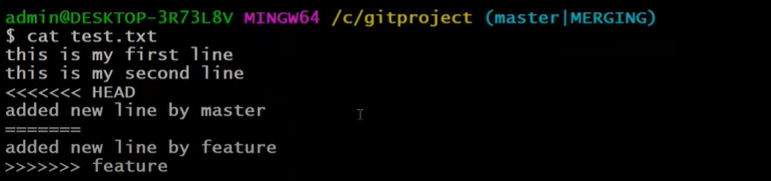




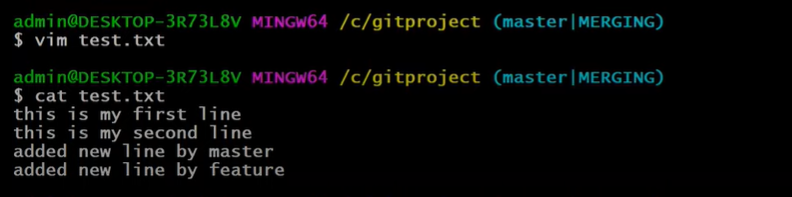




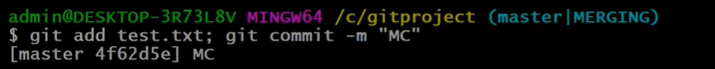
Now check the file that have conflict



This shows the common lines in the top  
changes in master and changes in the feature , which is causing merge conflict



In the above you open the file and make the necessary changes by choosing what needs to be in the file and save it

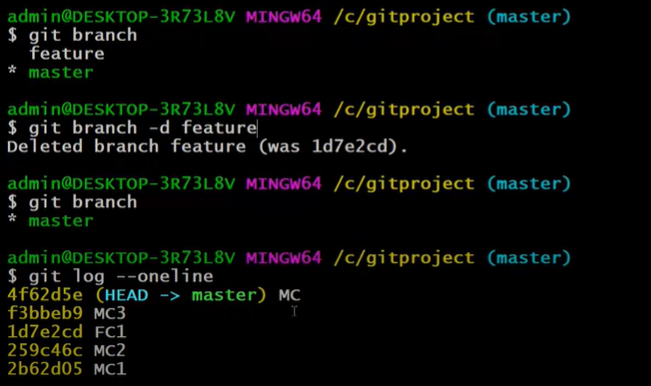


Committed the merge commit



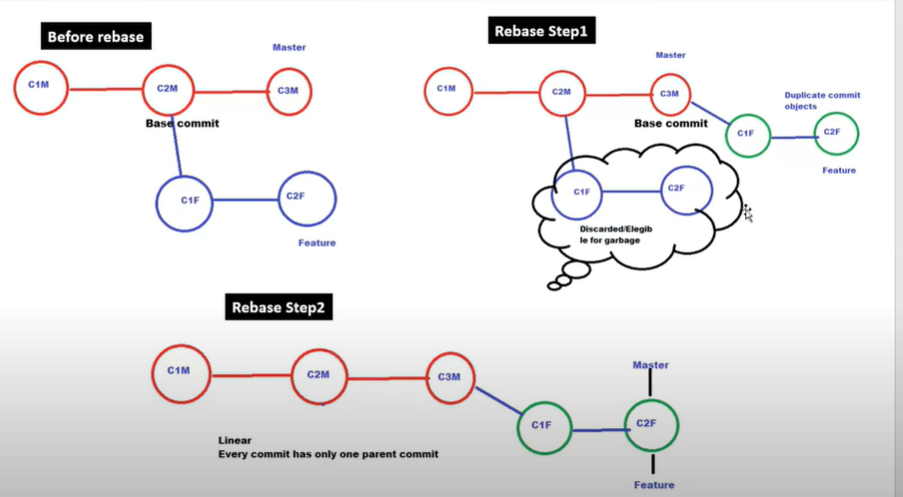


Once the work with feature branch is done, delete it as below



## Git Rebase

Note: this should be done on local repo and not on remote

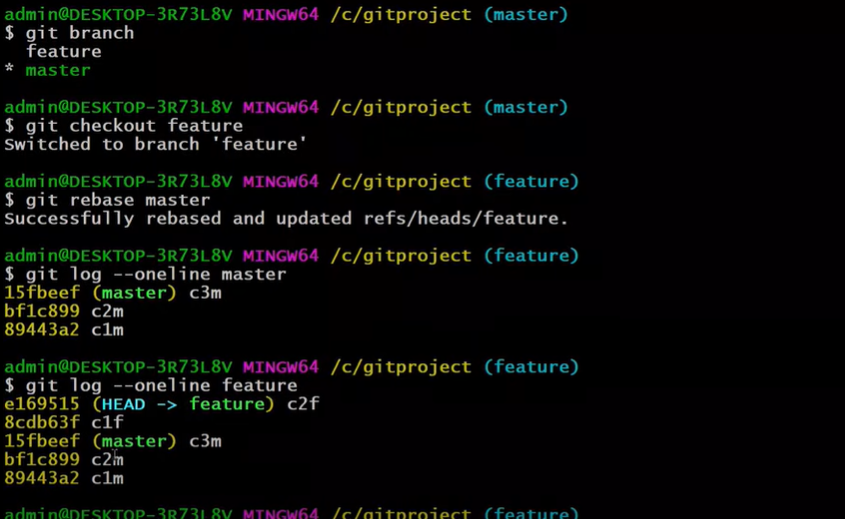


Here the feature commits will be made as virtual commits and all the commit id’s related to feature are deleted but feature commits and its messages will be shown in the commit log with different commit id

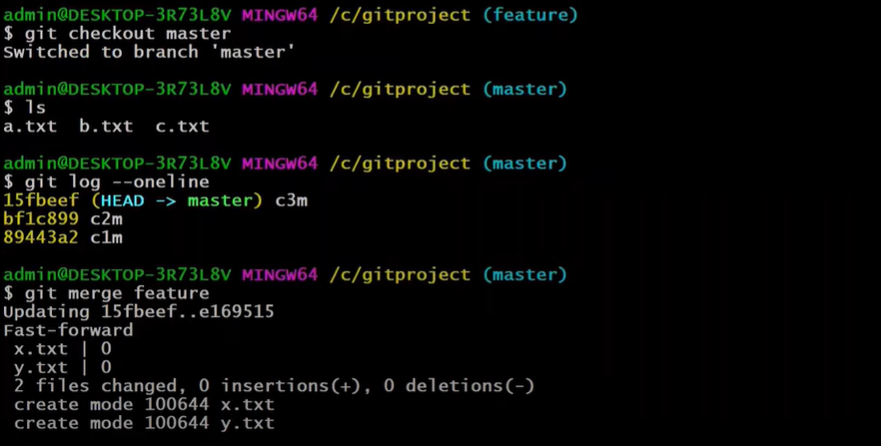
$git checkout feature

$git rebase master

Now feature commits are made on master branch



Now switch to master branch and commit



## Git Clone

Create a folder in the local machine

$git clone gitremoteurl

Example: $git clone https://github.com/vikramkvr/seleniumassignment.git

## Git Push

$git push origin master/main

## Git pull

$git pull origin main/master

## Git with IDE

In eclipse, import project from GIT—Projects from Git—Clone URI—next—copy the URI—Next—Import of general project.

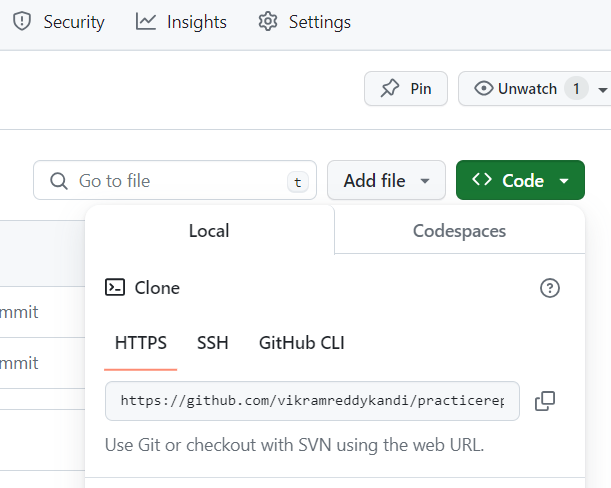
Select the project and right click, under Teams option you can find all the Git options

# Personal Execution

## Cloning a repo/project

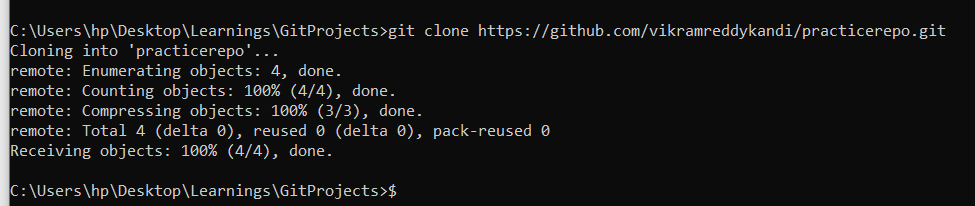
In command prompt Navigate to the folder where you want to get the cloned repo

Copy the https URL from the github repo



In the command prompt from the directory enter the below command

*git clone https://github.com/vikramreddykandi/practicerepo.git*



## Working on the repo

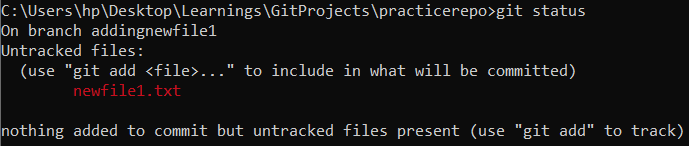
Create a branch, when you want to make changes to the existing repo

*git checkout -b addingnewfile1*

Now add a new file to the repo with some code/comments

Now check for the changes with below command

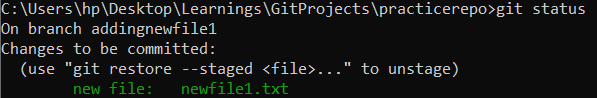
*git status*



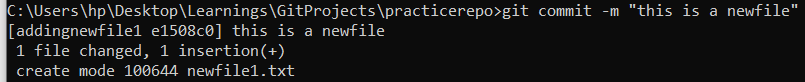
Now, to make the changes reflect, add the file so that those will be tracked

*git add "newfile1.txt"*

Now, do git status again to see in which state the changes are in



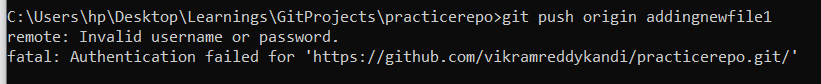
*git commit -m "this is a newfile"*



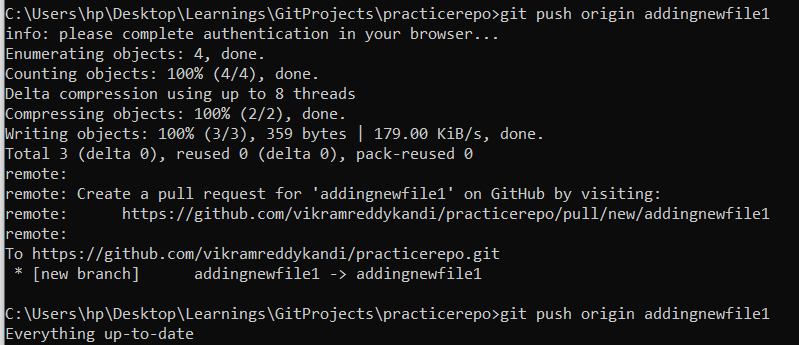
Now, push the changes as a new branch to github

Git push origin “branchname”

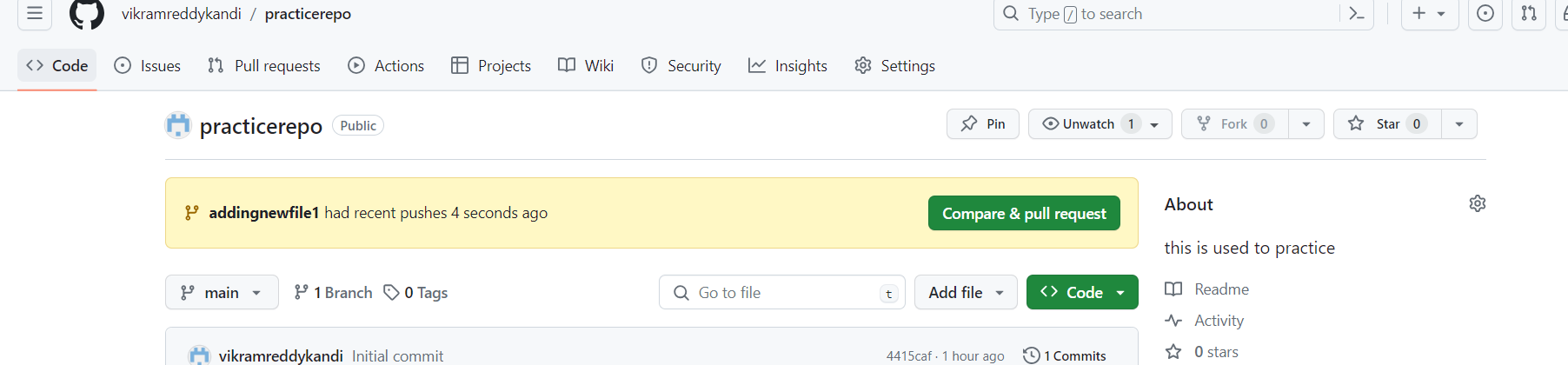
*git push origin addingnewfile1*



If you are using it for the first time you need to authenticate your self , after the above command a browser pop-up comes , which asks you to authenticate. You can use Personal access tokens or login using browser to github.



You will see something like the below in github, once you have pushed the local branch to git



## Conflicts:

After making your changes, use the below command to see whether there are any changes in git

*git pull origin main*

if there is a conflict, you will see the below

***Auto-merging file.txt***

***CONFLICT (content): Merge conflict in file.txt***

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

Now switch to main branch and pull the latest changes

*git checkout main*

*git pull origin main*

now to know about the changes, use the below command

*git diff ‘branchname’*

*git log –oneline* (this will give commits done in the current branch )

## Resolving Merge Conflicts from IntelliJ

After updating your changes in branch, assume that there is a push happen in master branch

Now when you try to pull the code changes from master, you will see a pop-up screen with title Conflicts

You will find the options ‘Accept yours’, Accept theirs’ and Merge.

Click on Merge, it opens a tool bar with three screens. Left and right will be local and master branches and middle one is the result editor which you can edit to resolve the conflict.

After resolving , you add, commit and push the changes to git

## Resolving Merge Conflicts from Eclipse

After updating your changes in branch, assume that there is a push happen in master branch

Now when you try to pull the code changes from master, you will see a pop-up screen with title Conflicts

Close that.

Go to the file which is causing conflicts, right click – Team—Merge Tool

Compare and update the file that you are going to commit and push.

## Updating Personal Access Token

In case when some the checkin’s fail due to permission issues, you can regenerate the token in git

In the upper-right corner of any page on GitHub, click your profile photo, then click Settings. In the left sidebar, click Developer settings. In the left sidebar, under Personal access tokens, click Tokens (classic).

After this go to the terminal and give the command

git remote remove origin

git remote add origin https://[NEWTOKEN]@github.com/[USER]/[REPOSITORY]

git push

then it will get reauthenticated with new token

## For Cloning existing project

echo "# Playwrightproject" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/vikramkvr/Playwrightproject.git

git push -u origin main

## for pushing existing project to GIT

git remote add origin https://github.com/vikramkvr/Playwrightproject.git

git branch -M main

git push -u origin main

## moving commits in Master to branch

when you have wrongly committed all the changes top master branch then use the below commands to move the commit # to a new branch or existing branch

get the commit # from master, use the below command

*git log*

create or checkout to the branch

*git checkout -b branch1*

*or*

*git checkout branch1*

now, provide the below command to move commit # to branch from master

*git cherry-pick commit#*

*git push origin branch1*

## Deleting Branches locally and remotely

// delete branch locally

git branch -d localBranchName

// delete branch remotely

git push origin --delete remoteBranchName