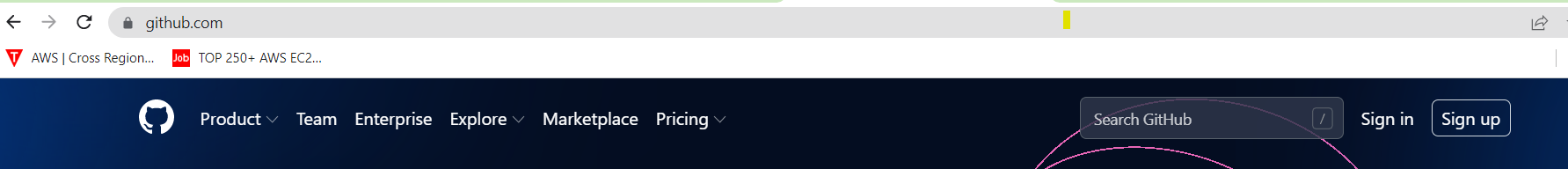
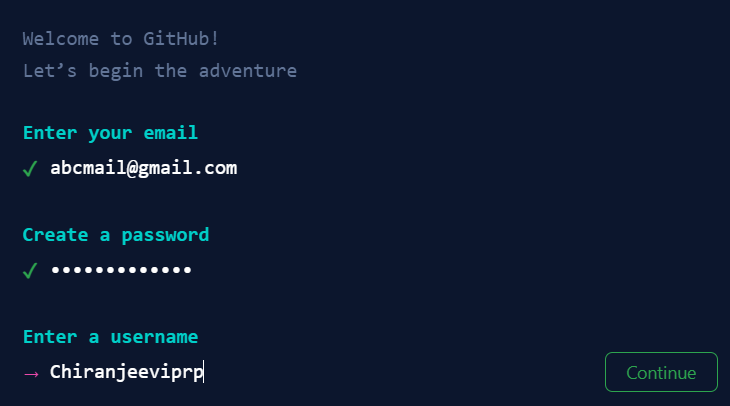
Step 1:- [www.github.com](http://www.github.com)

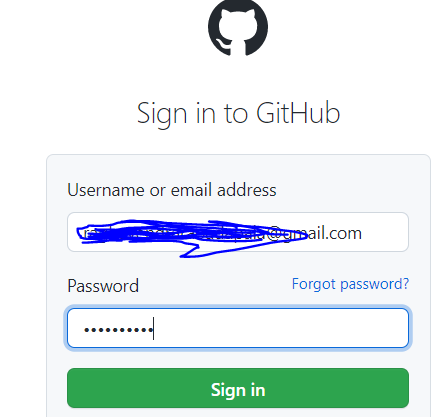


To create github account🡺sign up

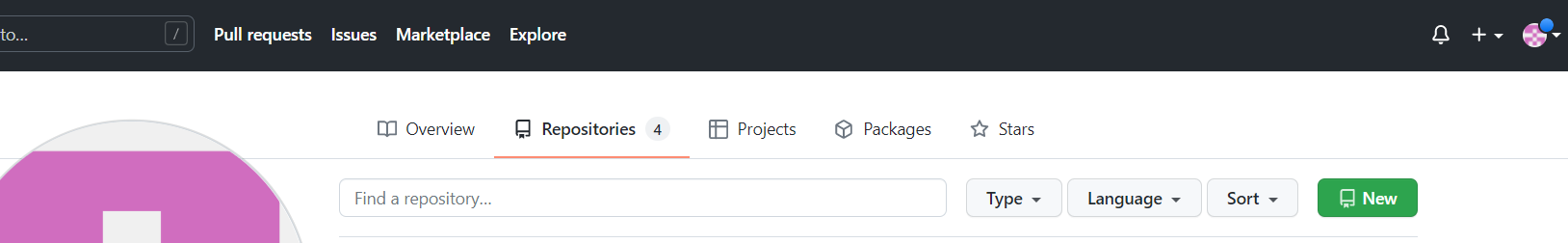


Enter your email and create password and create username then continue and solve puzzle to create github account.

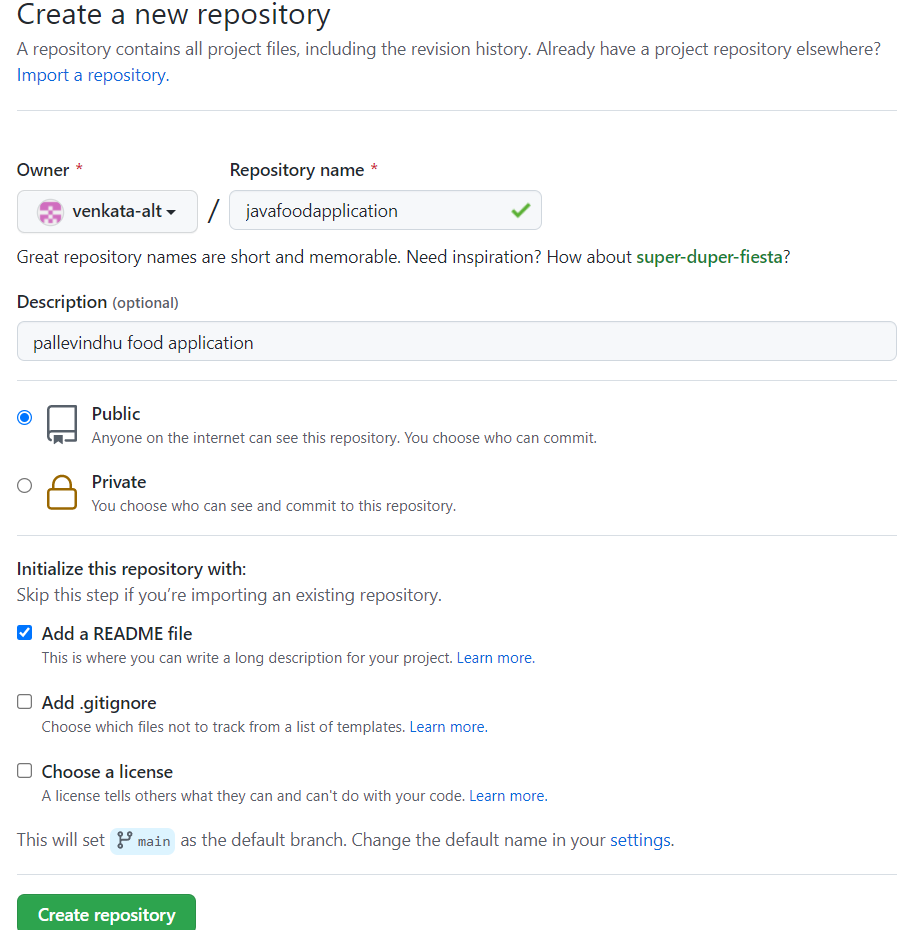
🡺sign in github account

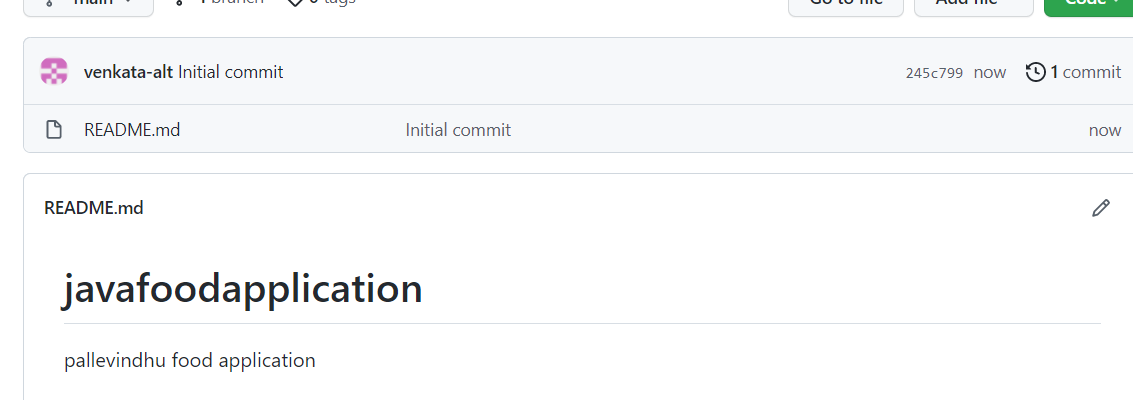


Step 2:- create repository (to go repository)



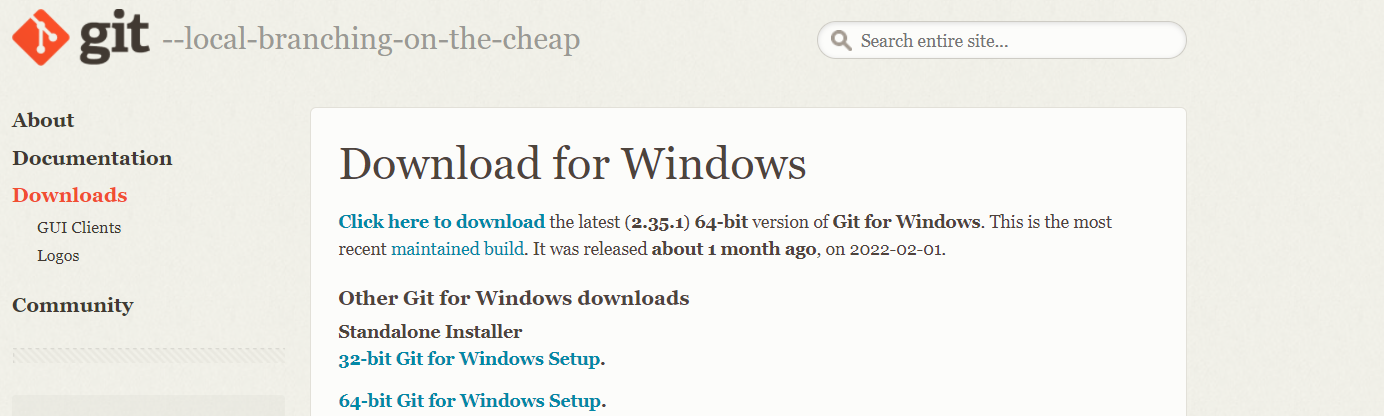
Select on new🡺to create a repository on below image steps



After creation repository to see below image 

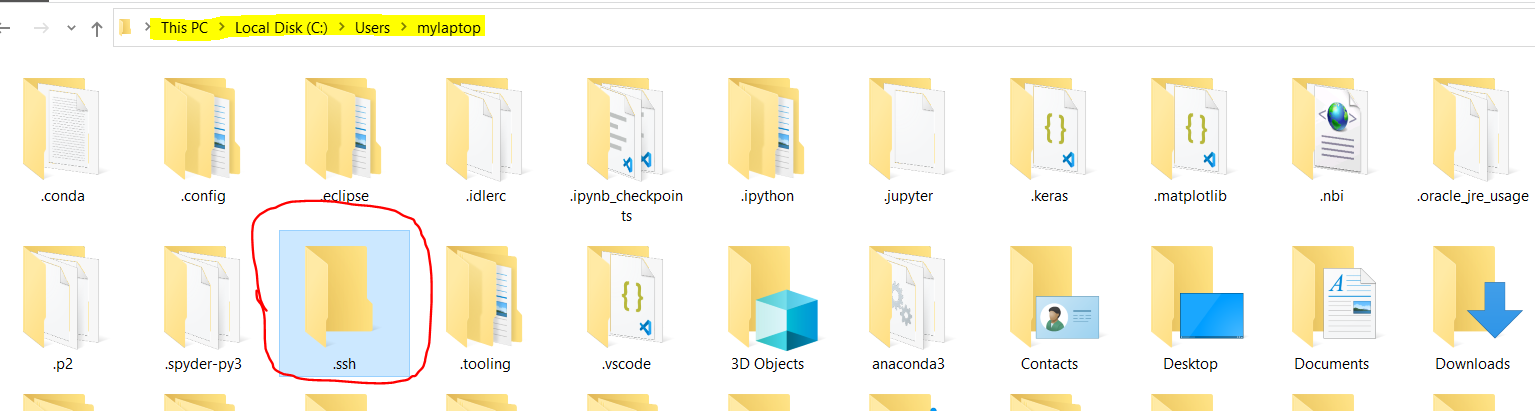
Readme:-readme file contains information about the repository, whatever info given in description will be placed in README.md file.

Step 3:- go to chrome and download Git 64 bit and install after downloaded.

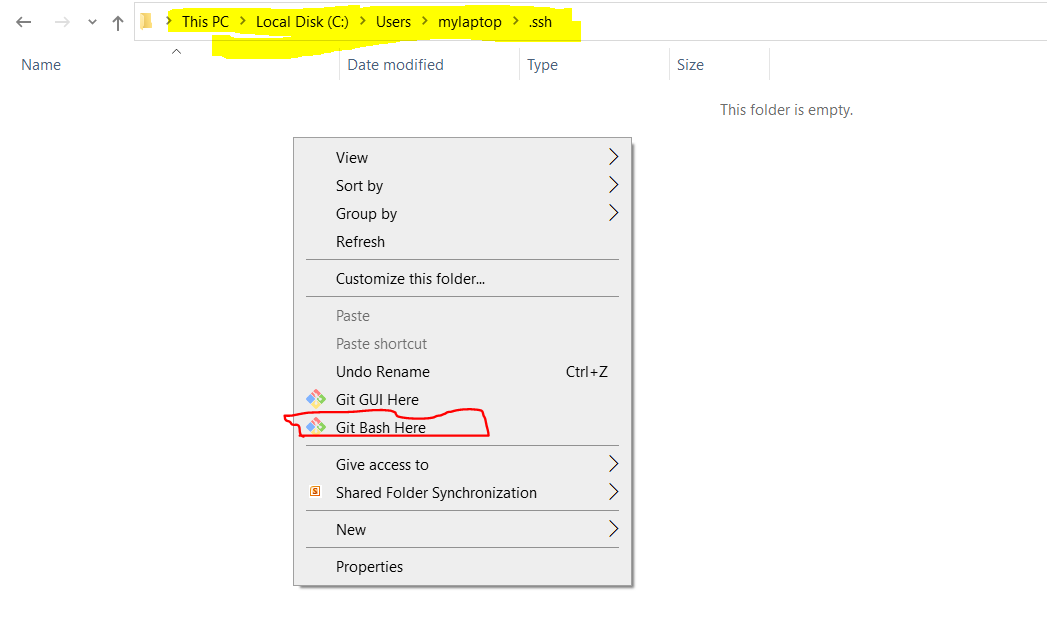


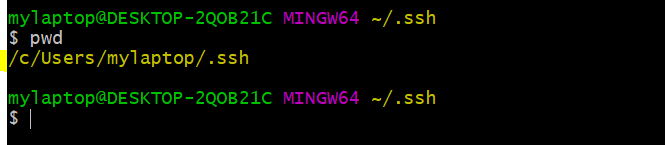
Now we have to create the ssh keys in the path “c:/user/<ur user name>/.ssh”

Open c drive 🡺 go to users folder🡺 go to your user name folder🡺 create a folder with name “.ssh”

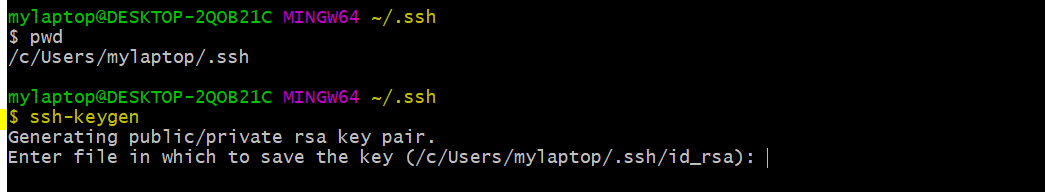


Now open the ssh folder and right click and select git bash here





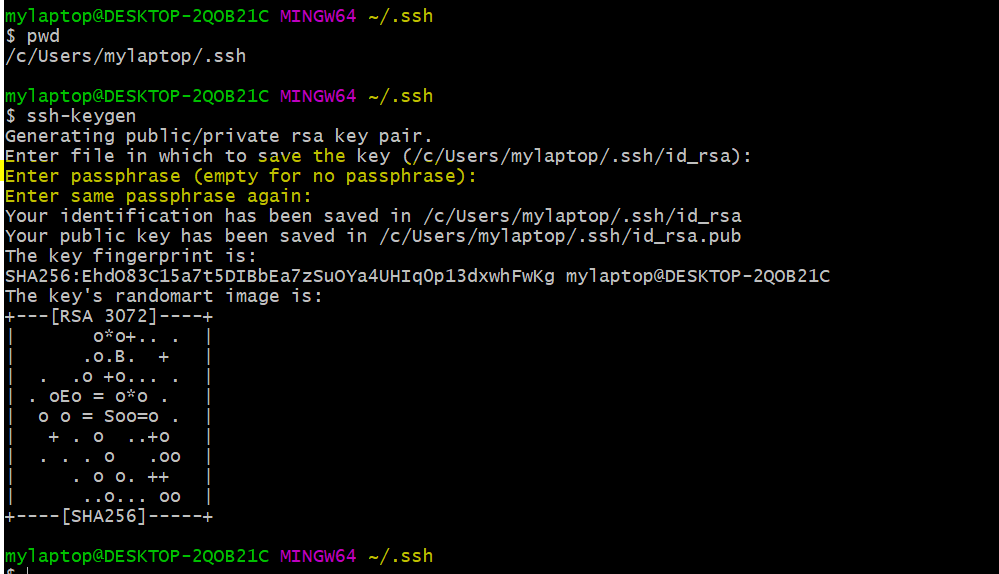
Now generate the ssh keys here



When we run the “ssh-keygen” command it will ask to enter the file name to save the key,if we don’t give any name by default it will take the file name as “id\_rsa”.

Next it will ask to enter the passphrase, just leave this as empty click enter.

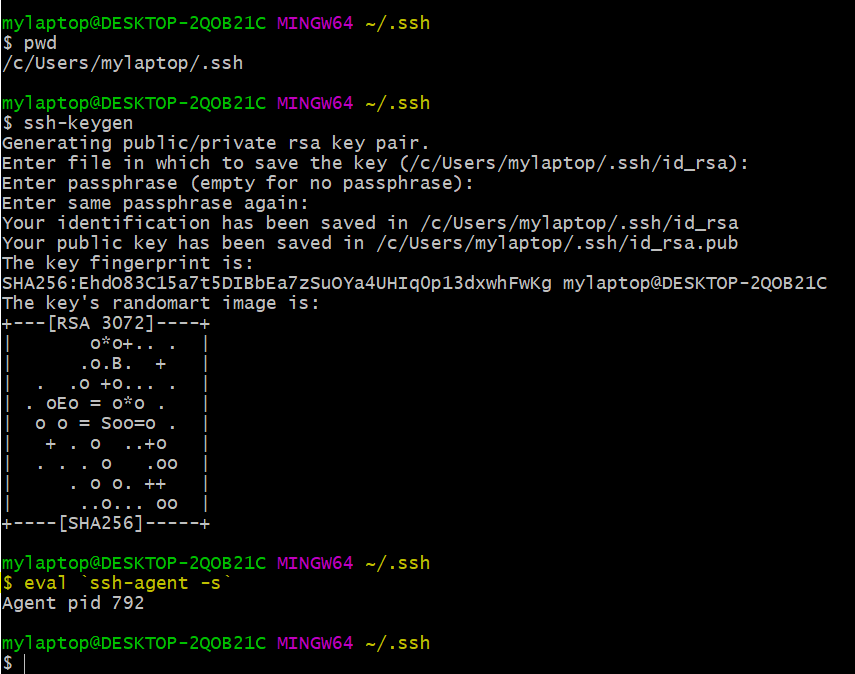
It will ask the one more time to confirm the passphrase, if u don’t given anything in the above step then here also don’t give any thing.this is like a password and confirm password so both times we have give the same passphrase.



> ssh-keygen #(to generate ssh key)

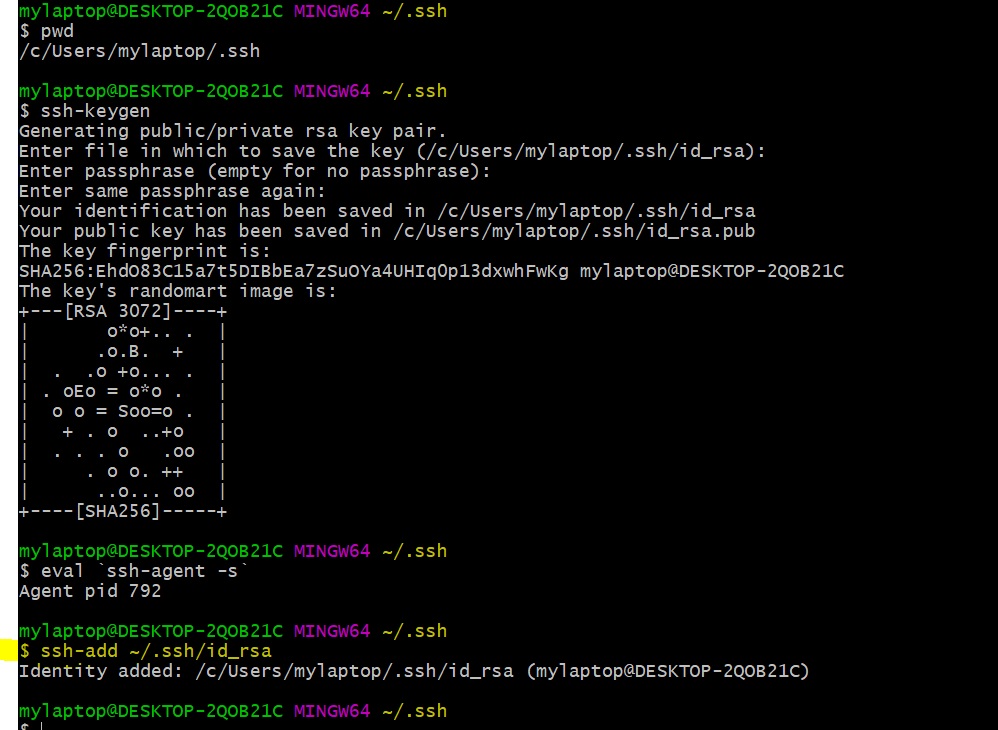
After creating the keys we need to create the agent

> eval `ssh-agent –s` #(to create agent)



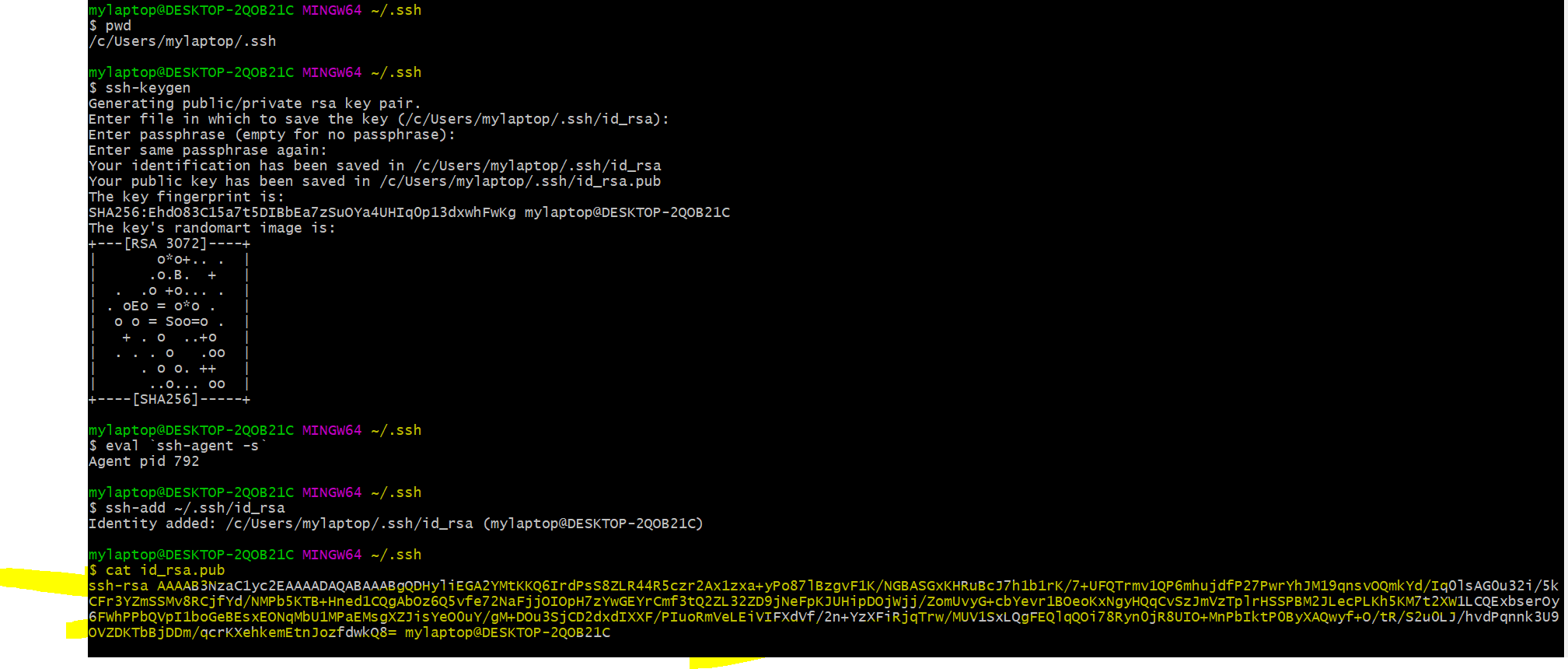
After creating the agent, add the privatekey to the git bash

> ssh-add ~/.ssh/id\_rsa #(to add the identity in git bash)

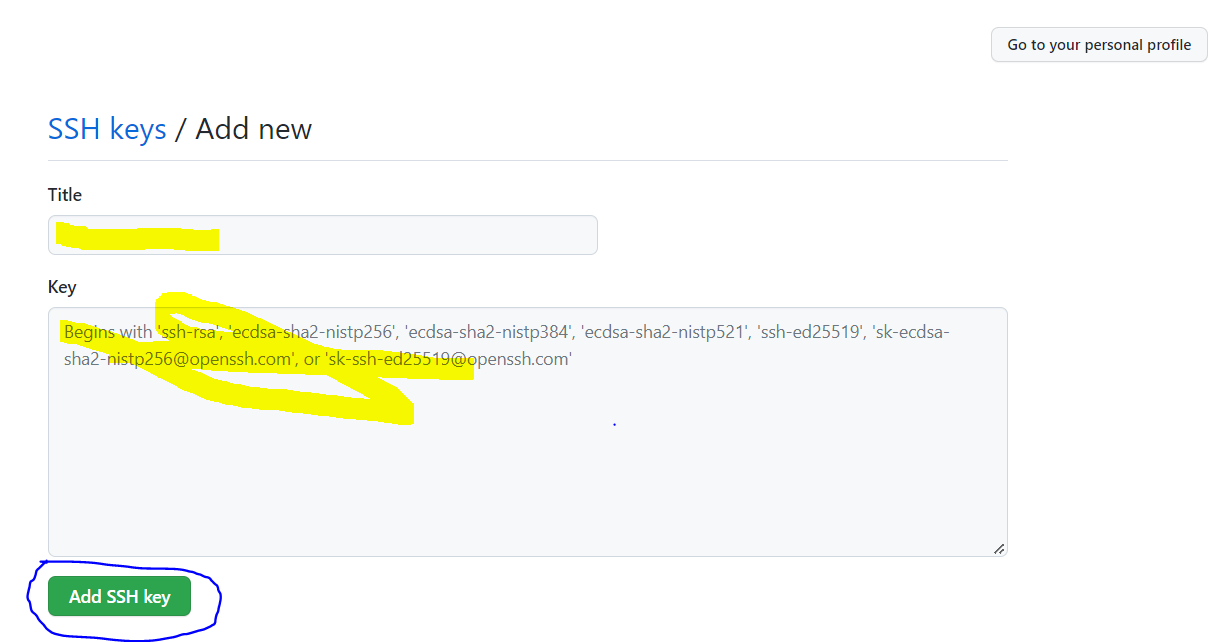


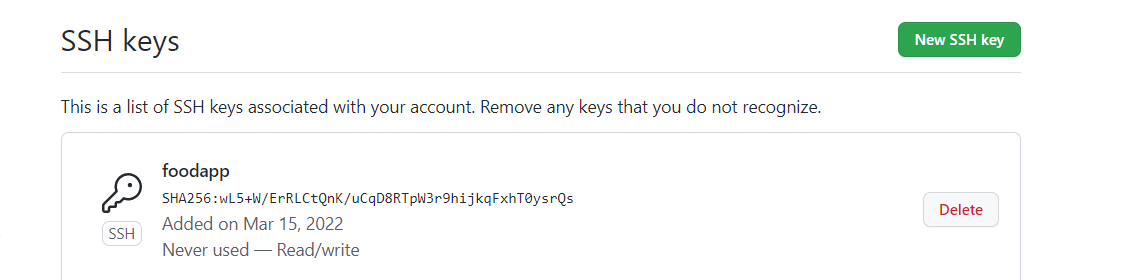
Now read the publickey using cat command

> cat id\_rsa.pub # ( to read the data in the given file)



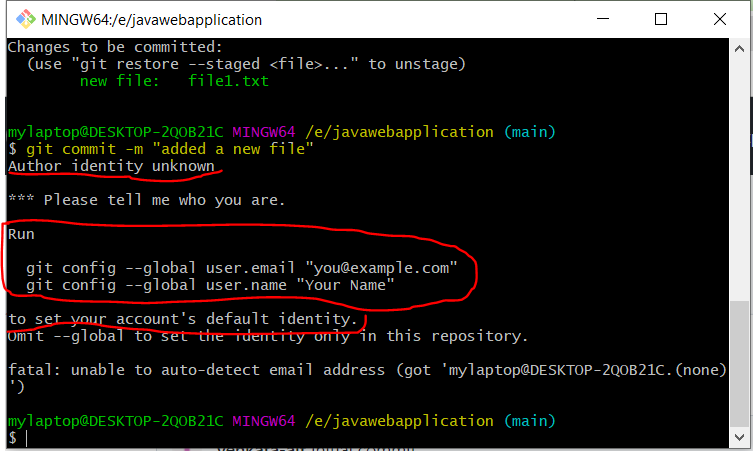
After that we go to 🡺github.com🡺account settings🡺ssh and gpg key🡺new ssh key🡺give tittle and paste the key🡺add ssh



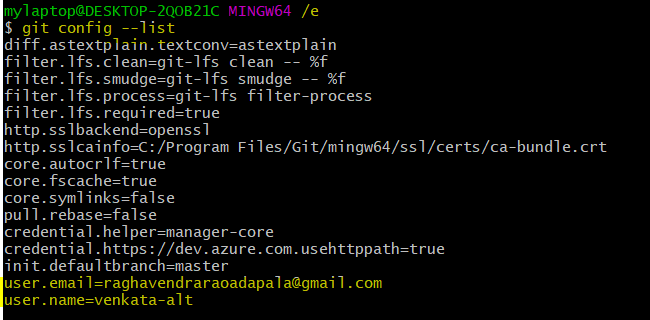


Now authentication between gitbash and git hub done.

After authentication is done, configure the email and user name in git bash.

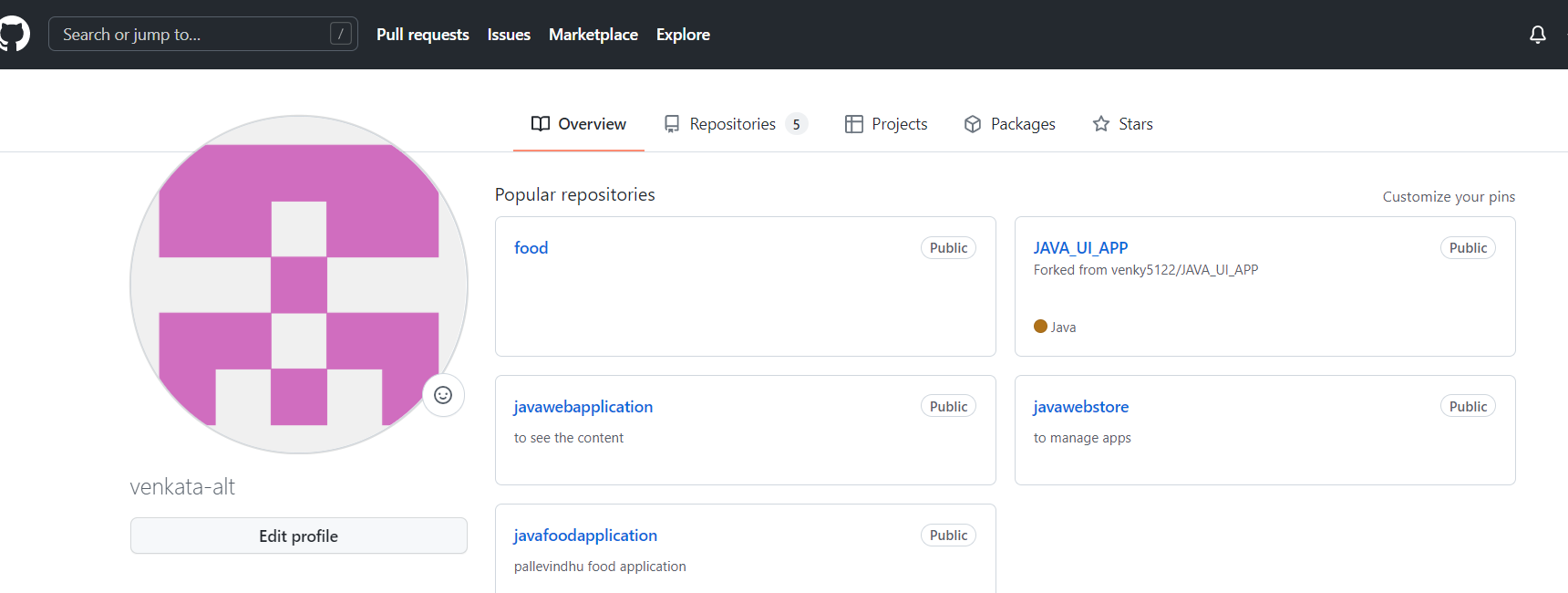


After configuration is done see if it’s updated or not in the config.



Class 2:-

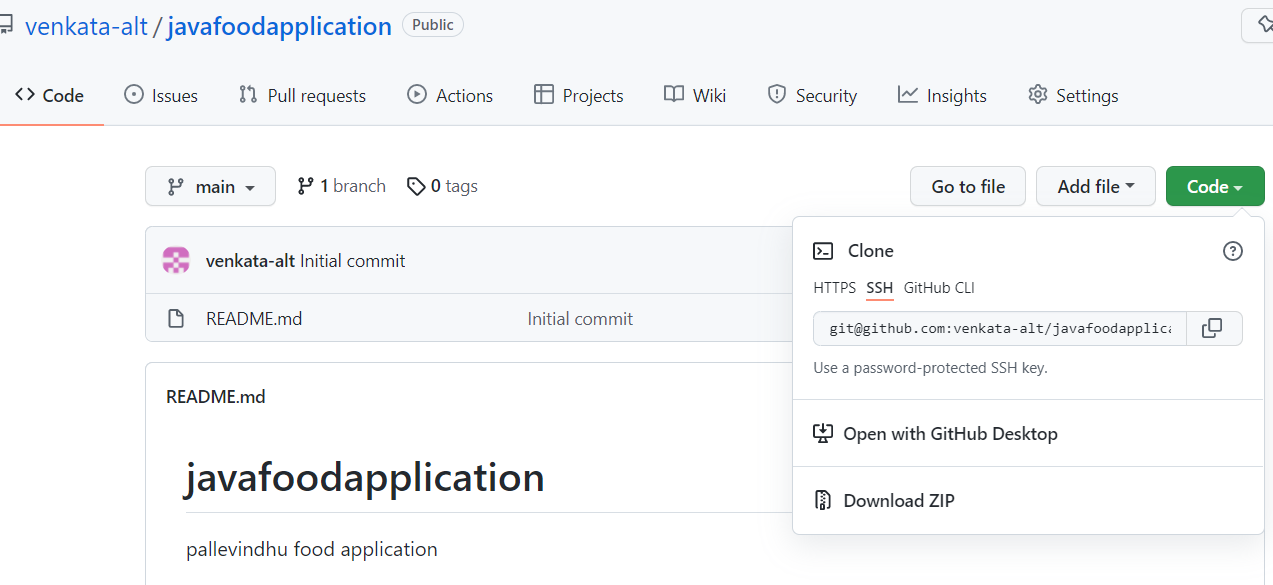
Go to git hub and select one repository



Clone:-how to clone central repository to local repository

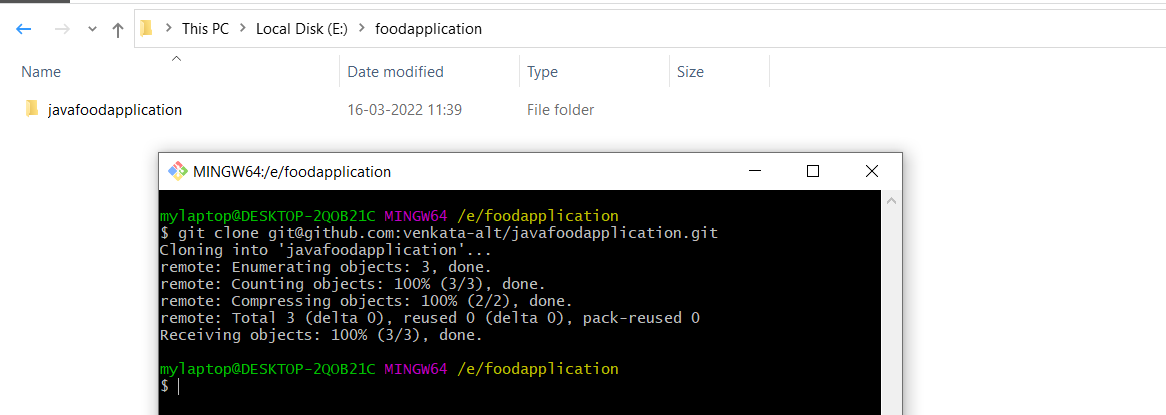
Command: - git clone “repository url”

Copy repository from food application to see below figure

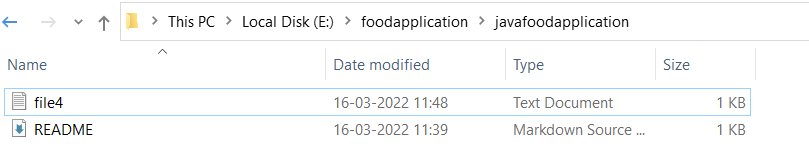


Open gitbash and select where you can copy clone repo

For example:-I am saving my repo in food application



I am adding one file in food application



By default when we make changes in the repository they are stored in cache or buffer.

How to check cache or buffer files

Command:-git status

Fatal: not a git repository (or any of thr parent directories):.git( it is they

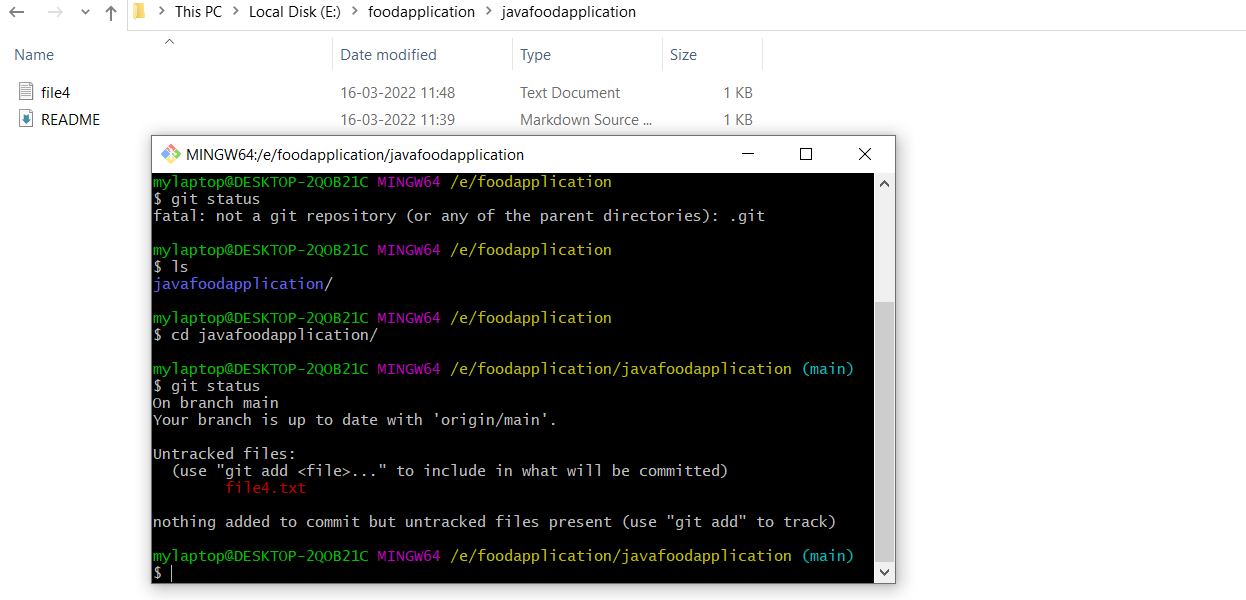
ls (to list the files)

Change the current working directory to cloned repository.

> cd <repository name>

Ex: cd java food application

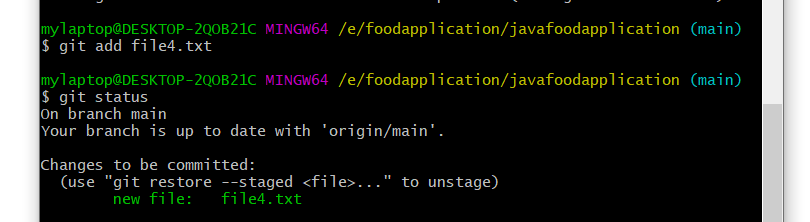
git status



How to add file name in stage area

Command:-git add <file name>

After changes added into staging area they will show green colour

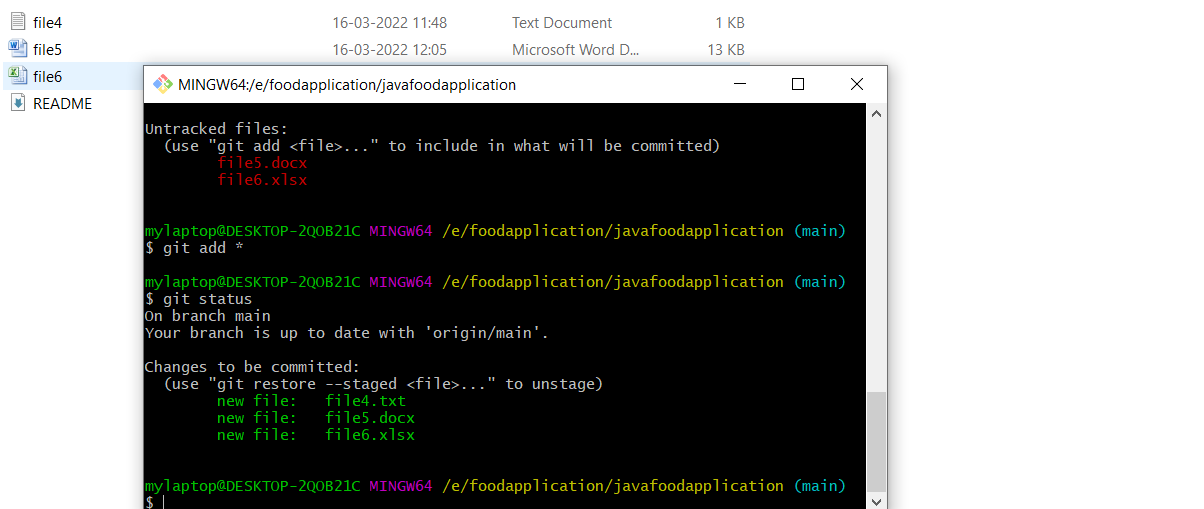


If u adding all files at a time to use this command:- git add \*

see below fig we are adding two more files,by default it’s stored in cache/buffer

After that we adding stage area command:- git add \*

Git status # to see the status

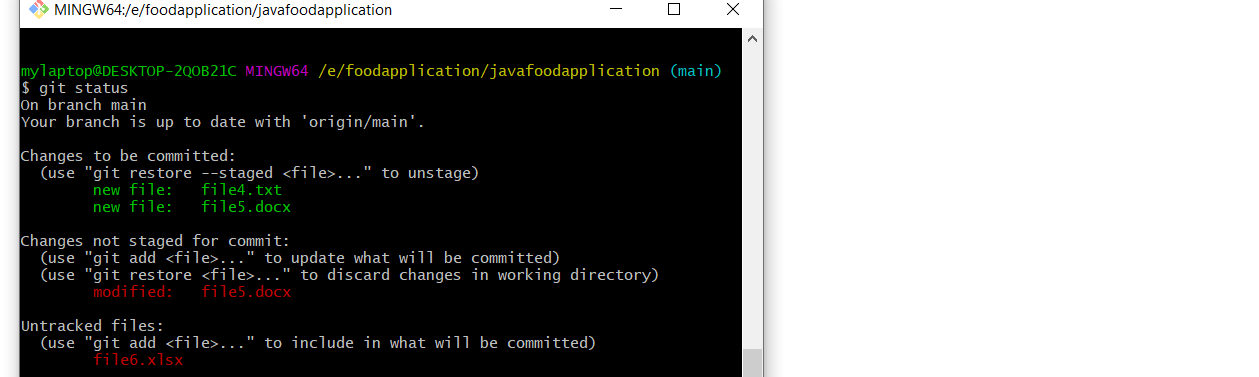


How to restore a file from staging area to cache/buffer by using this command:- git restore --staged <file name>

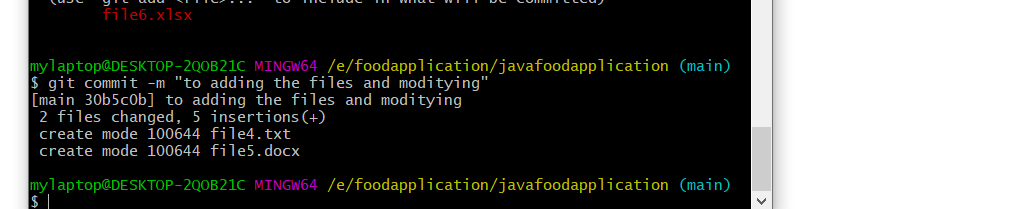


If you modifying in one file to see changes in files

git status

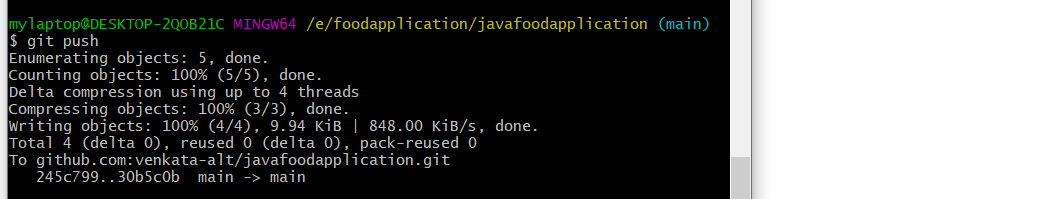
How to commit file stage area to local repo

Commad:- git commit –m “message”

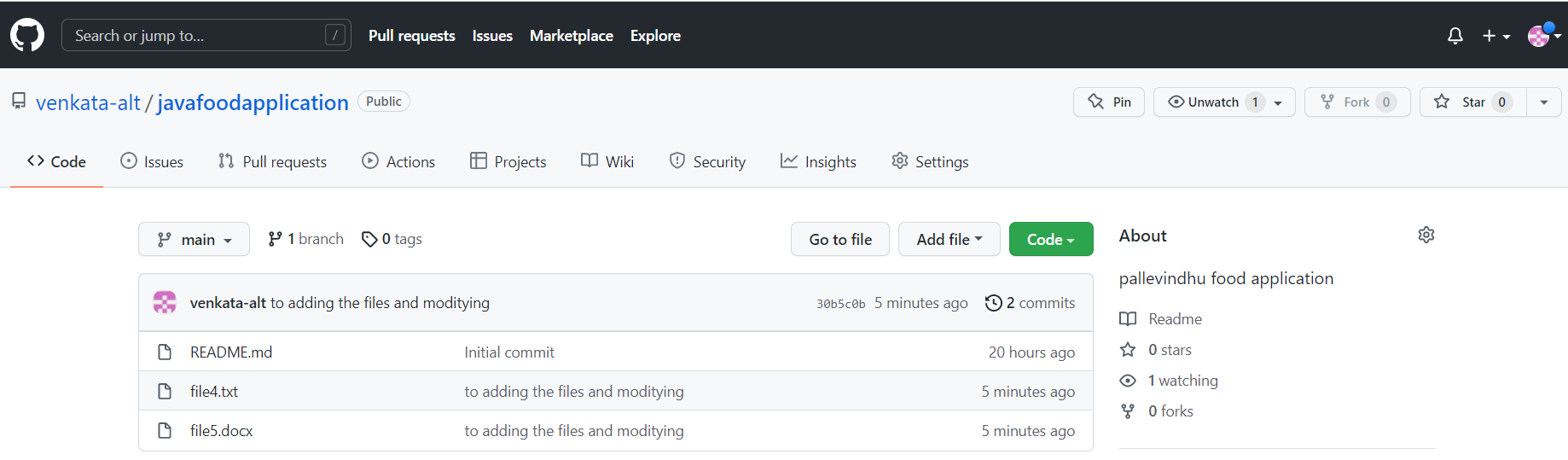


How to push the file from local repo to central repo

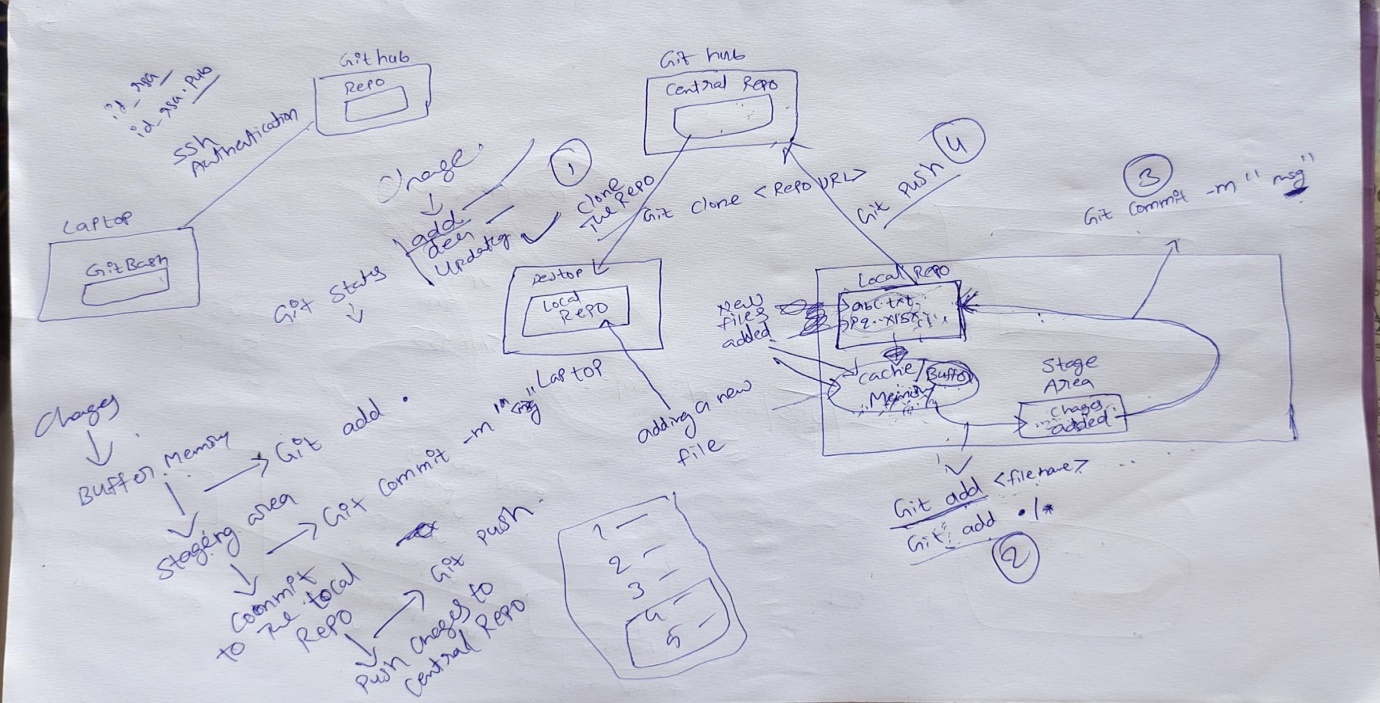
git push # to push the files



To check the central repository ,its adding the files from local repo to central repo

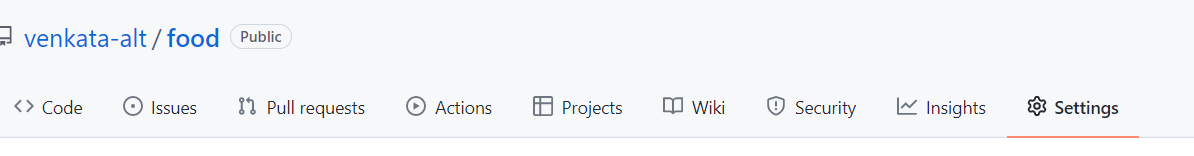


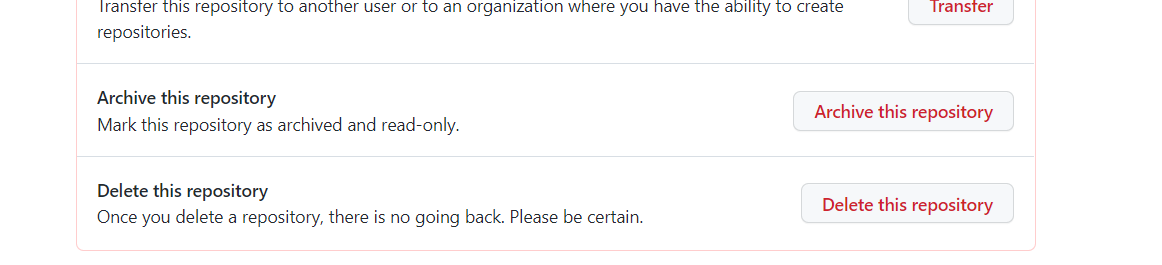
Class 2 demo :-



How to delete git repository:-

Go to repository settings 🡺scroll down 🡺delete repository

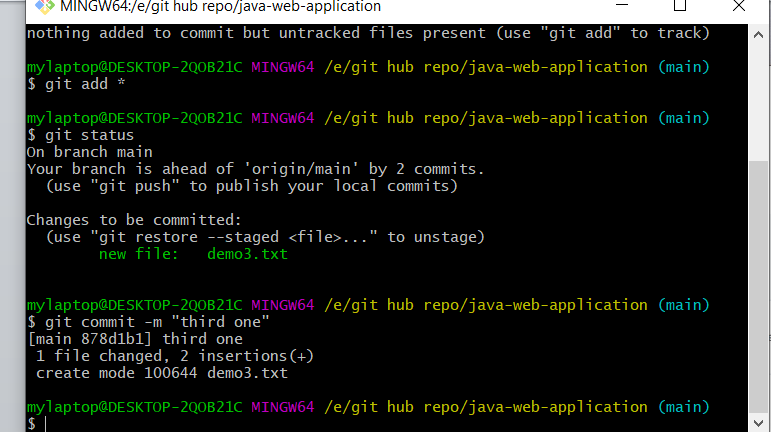




Class 3:-

How to restore commit changes to staging area after committing into local repo.

First we need to commit the changes on local repo

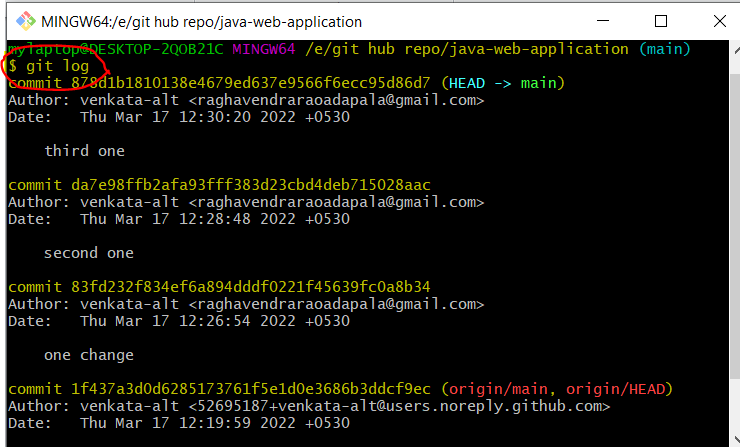


After that how to reback one commit

Using this command: - git reset <command id>

How to see commit id by using this command: - git log

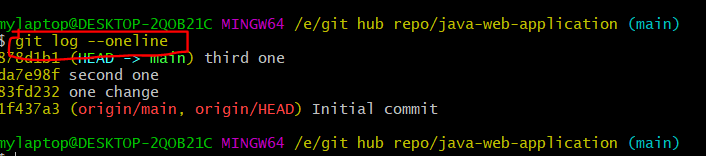
Git log is a command used in git to access the history of commits. its a brief history on commits orders



1. **Commit Hash**: The first part of the commit log is "**commit hash**" which is the hash value by which Git saves or refers everything internally. Refer to the Dot Git folder to know more.
2. **Commit Author**: This part tells you about who committed the changes in the repository i.e. the author name.
3. **Commit Date**: This part shows you the date and time of the commit with the day and time zone. It is important to have a day and time zone because many people work on a single project from different geographical locations.
4. **Commit Message**: This part shows the commit message that was written while committing the changes.

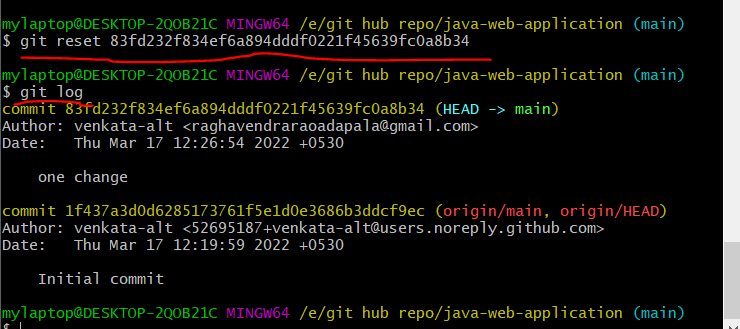
If you see one line commit:- git log –oneline

you can see some of the details that were previously visible are now hidden and the view is of only one-liner.

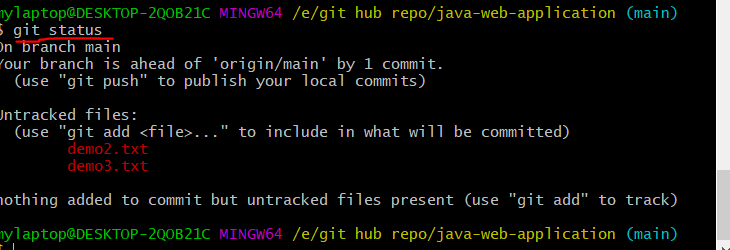


How to restore commit in stating area

Git restore <commit id>



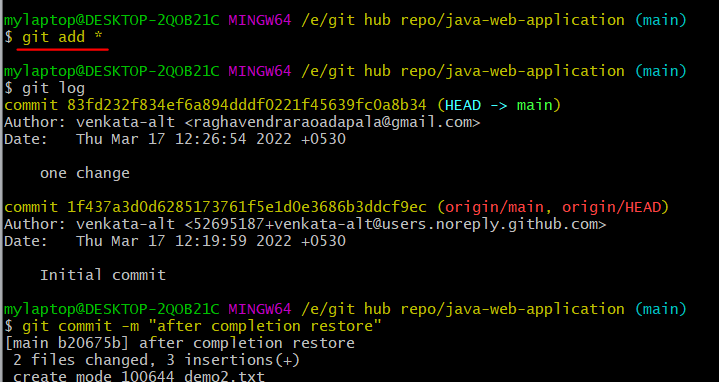
After this commit success check on git status:- git status



To add cache files on stage area after adding to commit local repository

Command :- git add \*

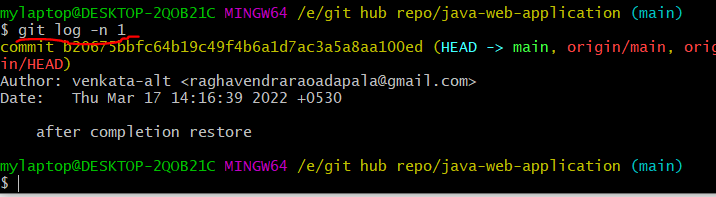
Command :- git commit –m “msg”



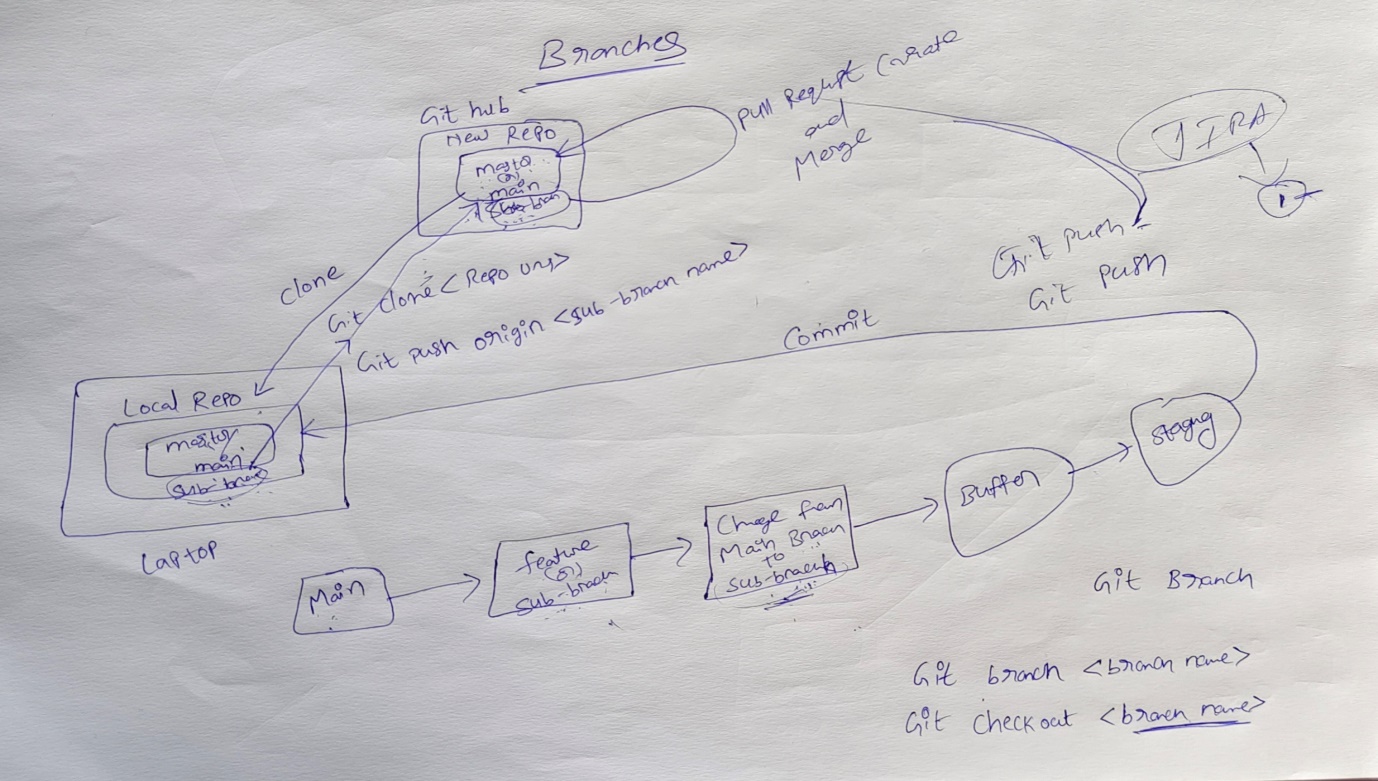
After completion of commit changes to push file local repo to central repo

Command :- git push

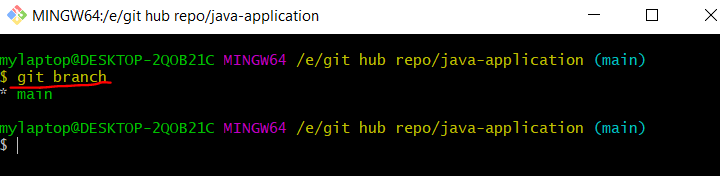
How to check top 5commits using this command:- git log –n <numbers>



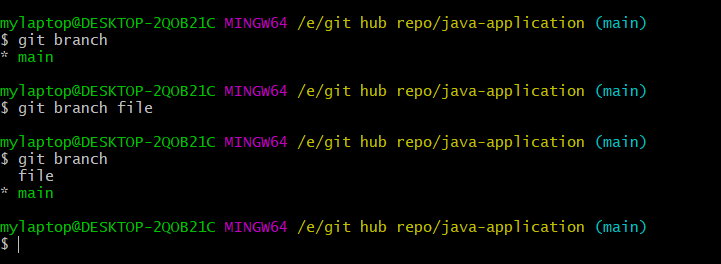
**Class 4:-**

****

As the above diagram we can understand the git repo 🡺 local repo🡺 after that changing directory 🡺 we can see branch name using this command :-git branch



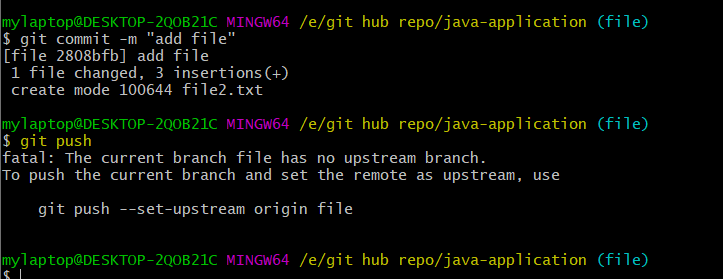
We can see git branch \*main 🡺by default it create a main branch🡺 in this main branch developer do not change the content because if one or more developers work on same app, so one dev doing a mistake entire code will be crash🡺 so every dev change a main branch to sub branch by using this command:- git branch <branch name>



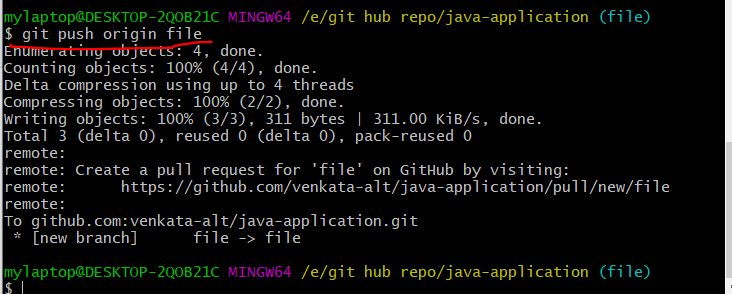
Main branch is green colour and sub branch as no colour 🡺 so we are changing from main branch to sub branch by using this command :- git checkout <branch name>



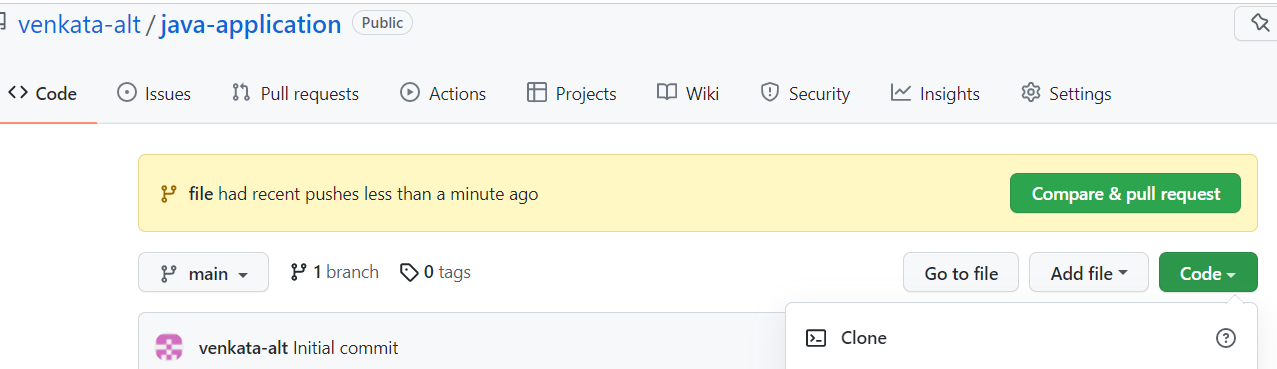
After changing branch 🡺we create a files on local repo🡺git add \*🡺git commit –m “msg” 🡺after that we push the code local repo to central repo 🡺 git push 🡺 its not push the file to central repo🡺because branch name is not matching



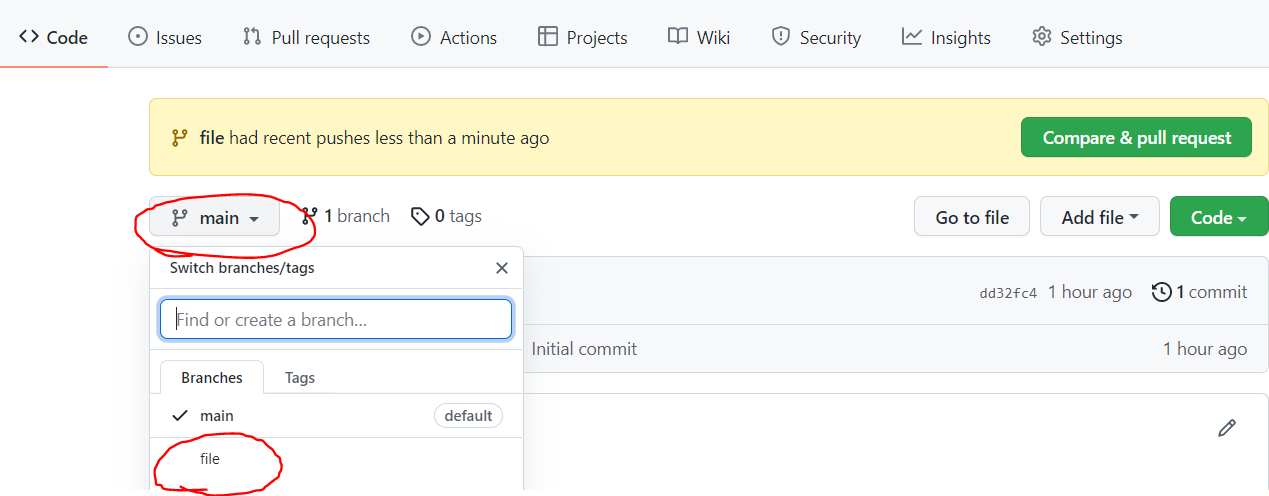
So after fail git push🡺 we push the code using this command:- git push origin <sub branch name>



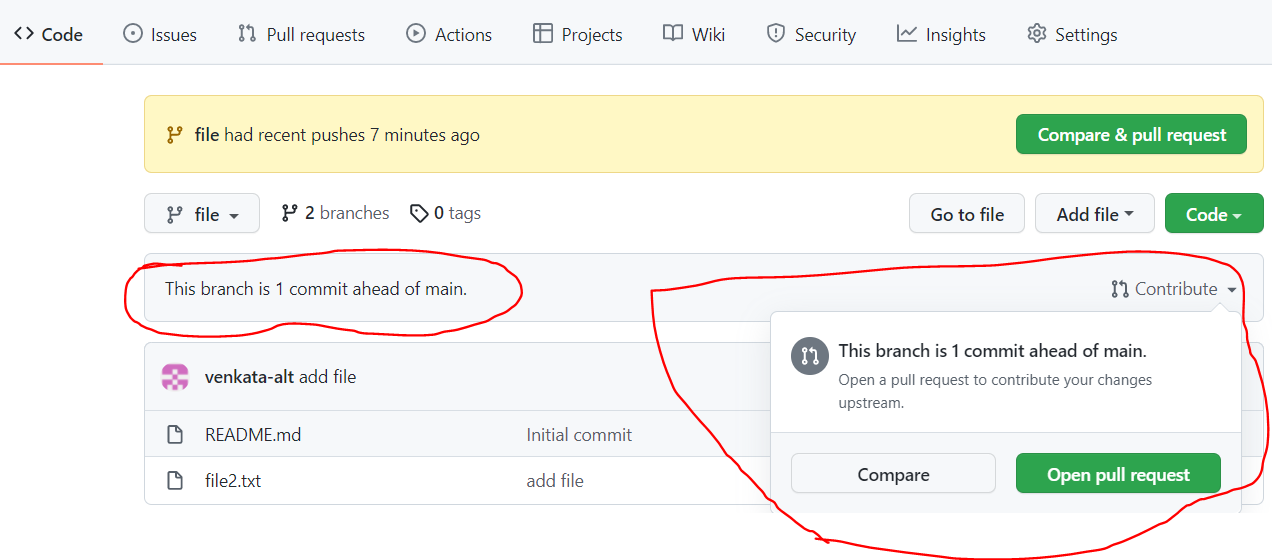
Go to central repo 🡺 waiting pull request



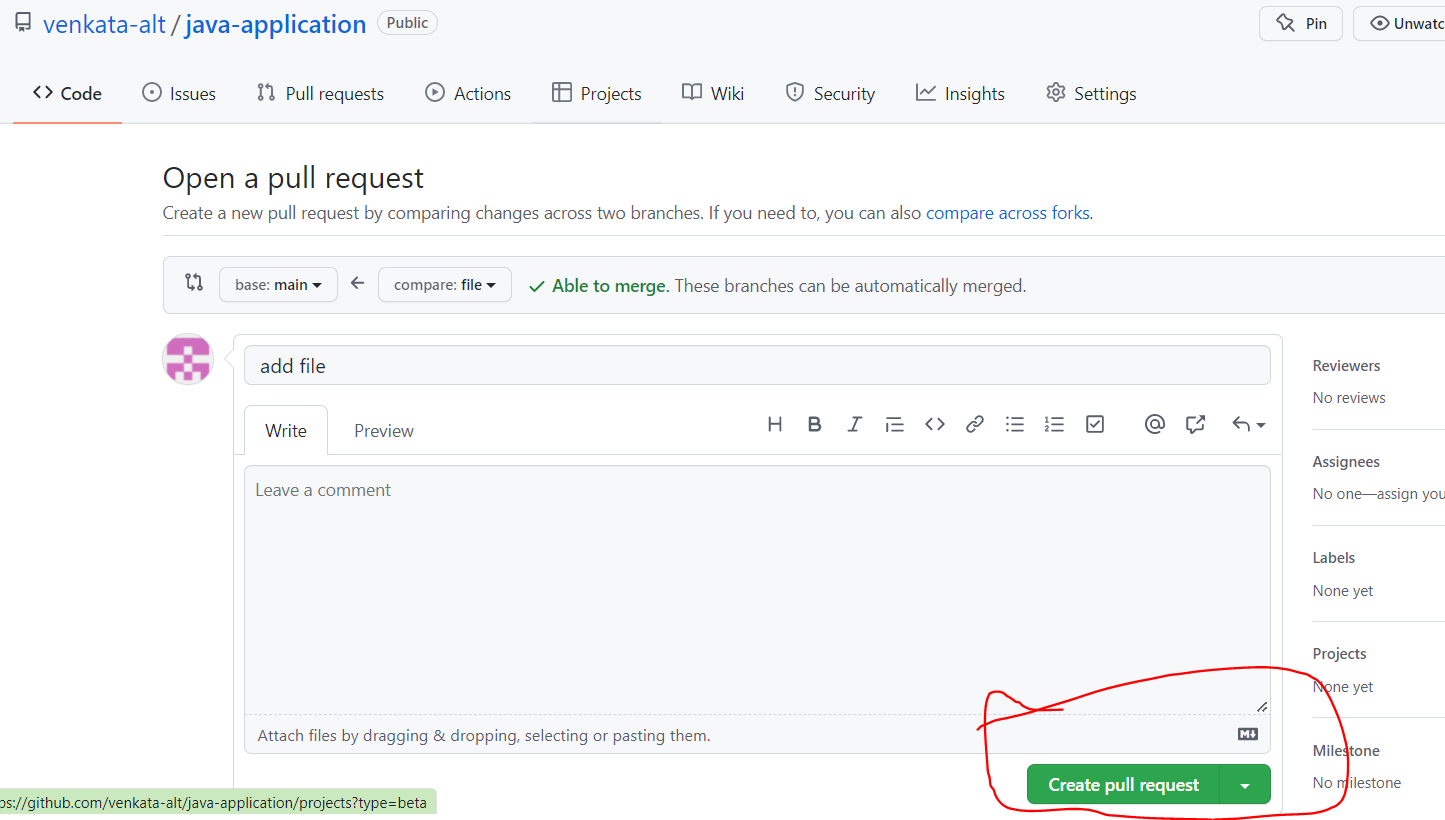
By default it waiting on when central repo is on 🡺 in case if they don’t shown anything 🡺 we go to branch select <created sub branch>



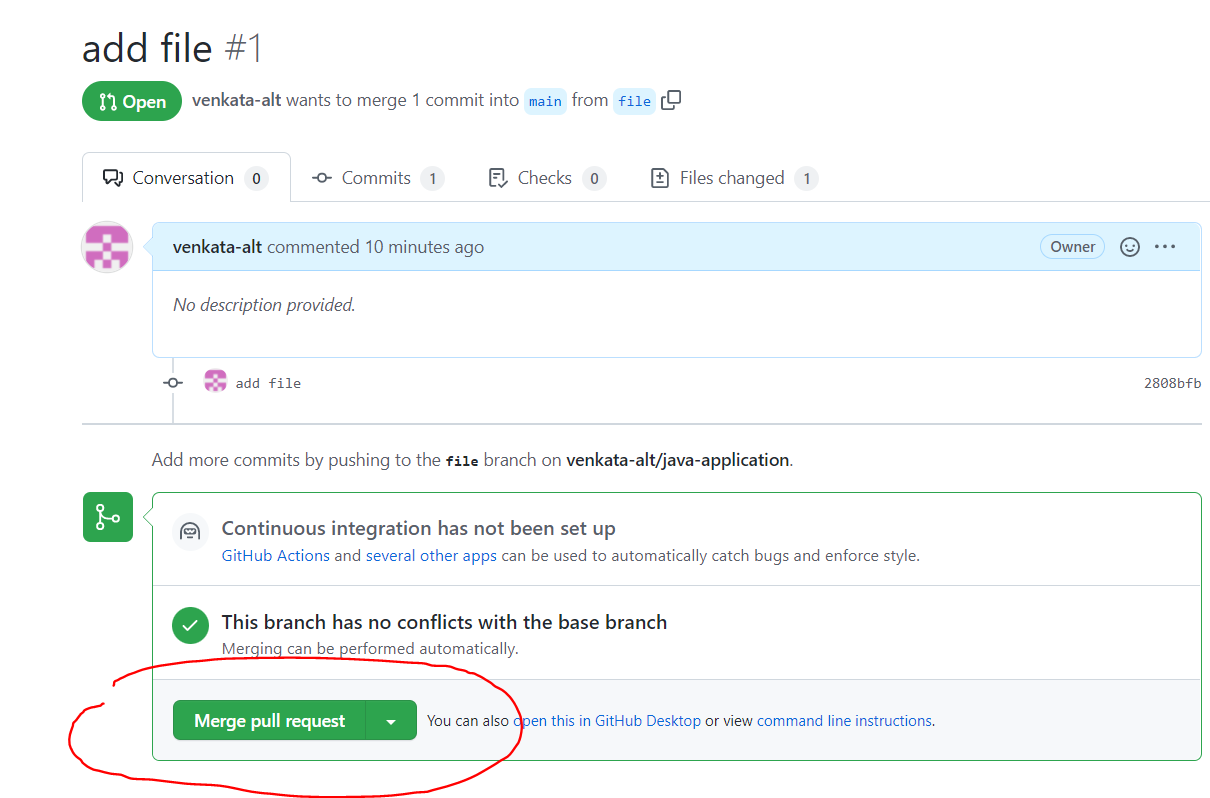
. After go to sub branch we can see pending request on contribute



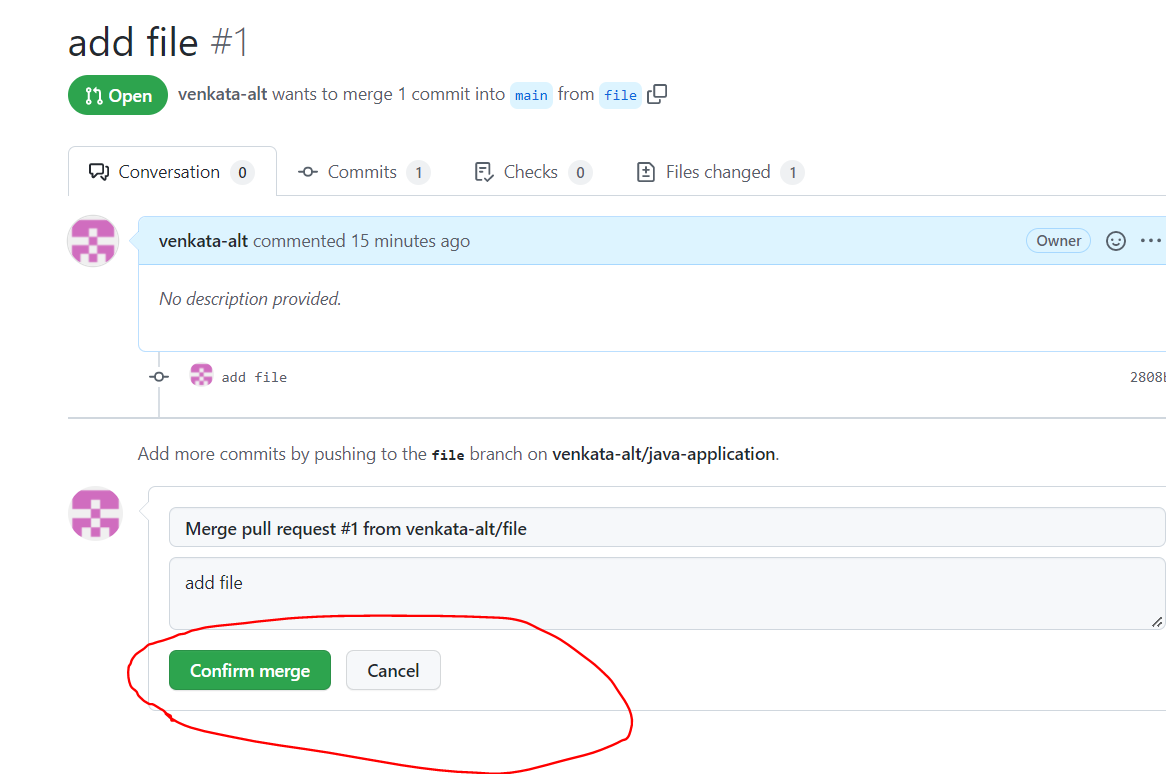
We can select compare & pull request🡺 after that we can click on create pull request



After create pull request 🡺merge pull request



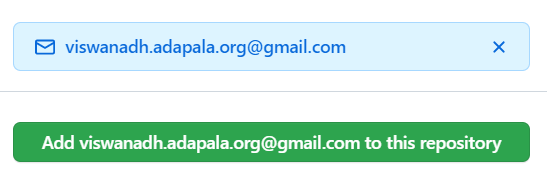
After merge pull request 🡺 confirm merge



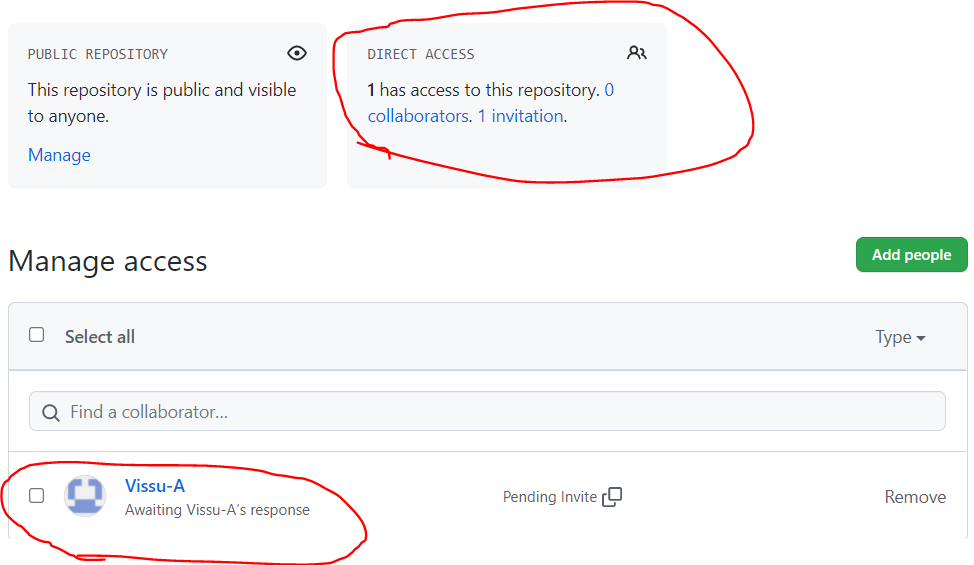
After completion confirm merge🡺 go to repo🡺refresh

**Class 5:-**

**Step 1:-collaborator adding repo**

Add a new collaborator 🡺 go to repo settings🡺click on collaborators🡺add people🡺give git hub mail id (other collaborators) ex:-viswanadh.adapala.org@gmail.com

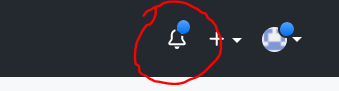
By default it has 0 collaborator after we adding 1 collaborator we can see 1 collaborator in below fig



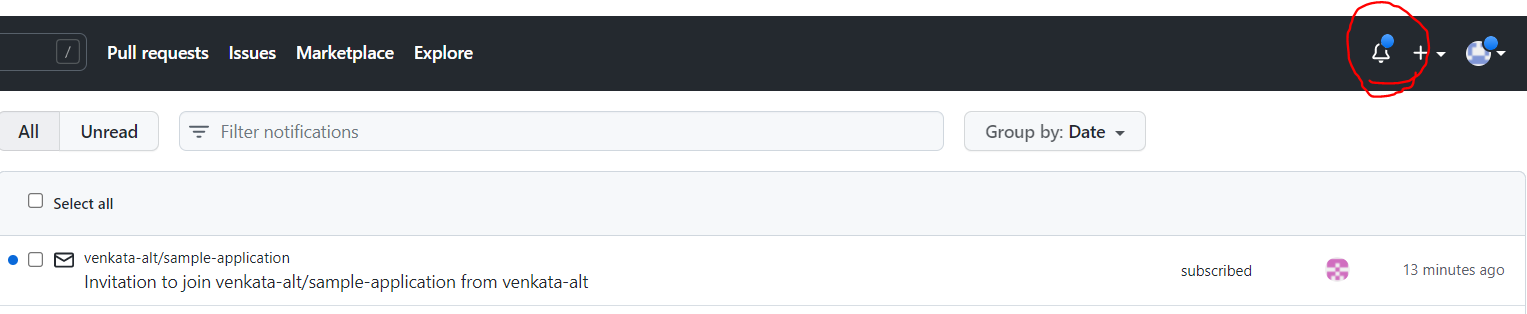
now we can see pending request on collaborator, after collaborator sign in to git hub



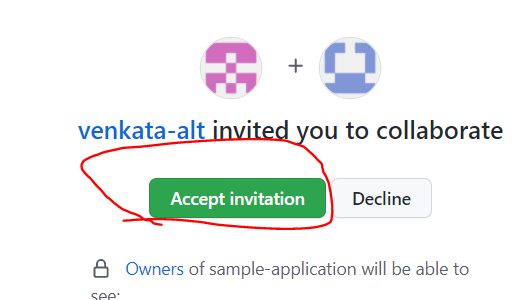
After sign in 🡺 collaborator seen notification



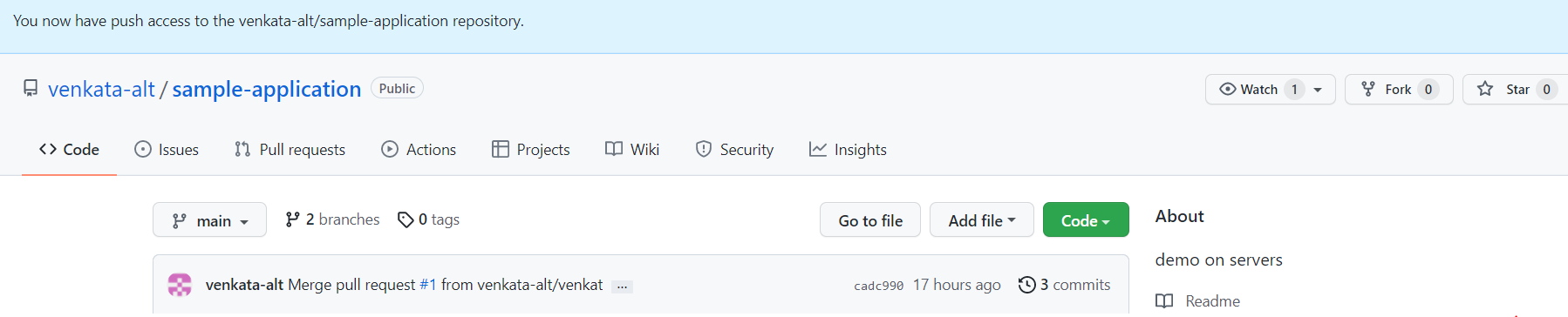
After click on notification🡺 now we can see one request



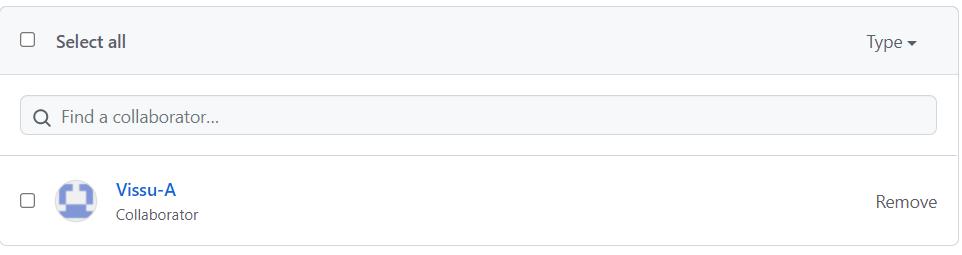
Click on that request 🡺now we accept invitation



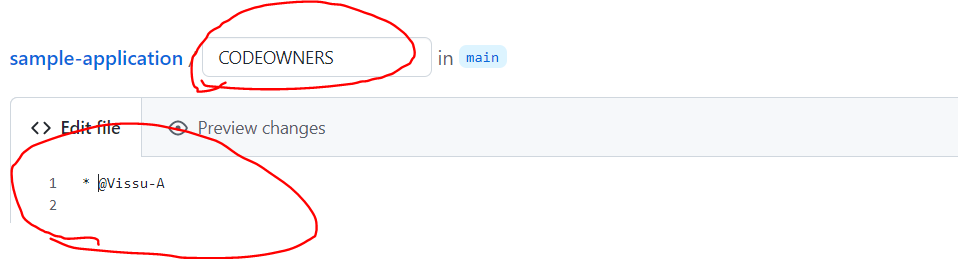
Now collaborator will have the push access to the repo



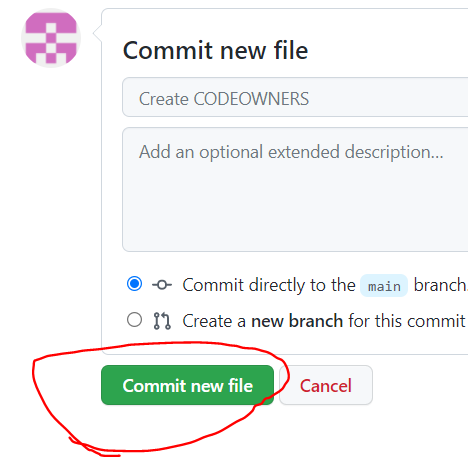
After collaborator accepting our request 🡺 we can see collaborator



**Step 2:- create new file for CODEOWNERS**

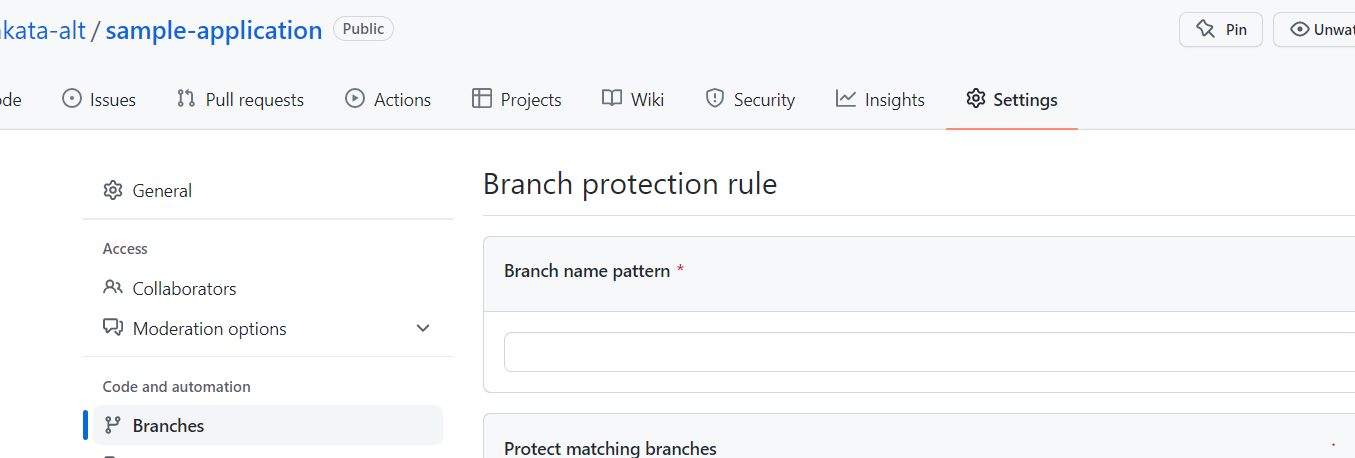
Go to repo🡺add file🡺create a new file with the name “CODEOWNERS”🡺 enter the code owner’s github user name (ex:-@Vissu-A)🡺we can add more than one codeowner

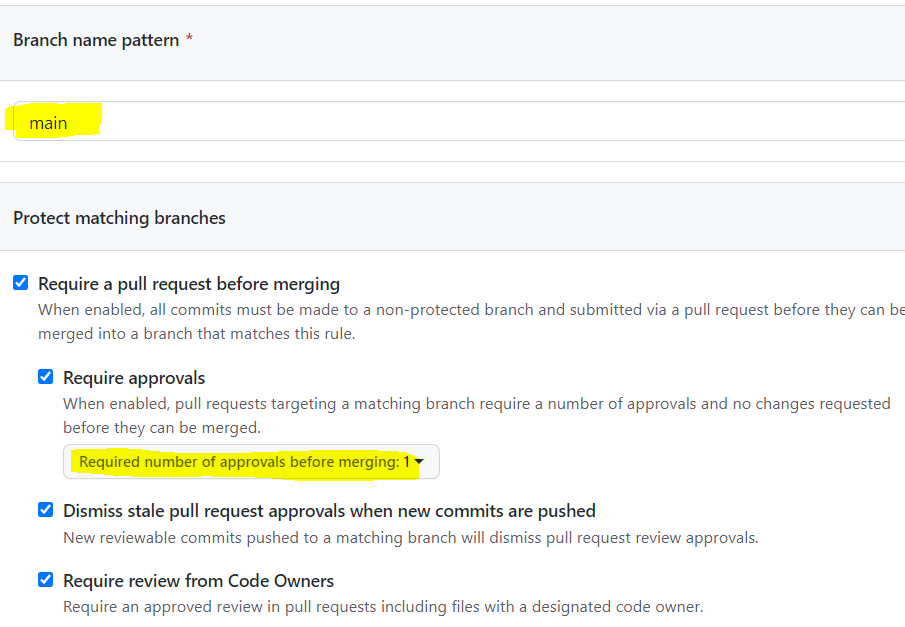
Scrolldown🡺commit new file



**Step 3:-Enable review required on main branch**

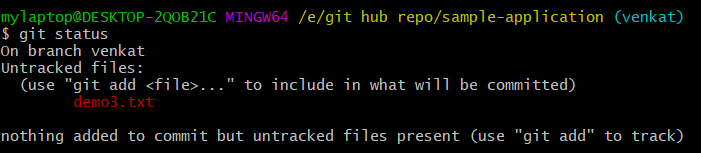
Go to repo settings🡺 click on branches🡺 click on add rule🡺 Branch name(main)🡺 click on require a pull request before merging🡺 click on require approvals 🡺now we can see one option(required number of approvals before merging) meaning is how many (collaborators)approvals accepting before merging🡺create

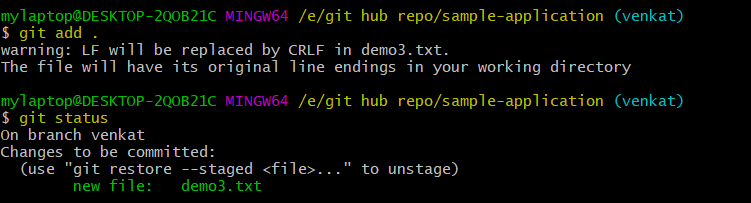


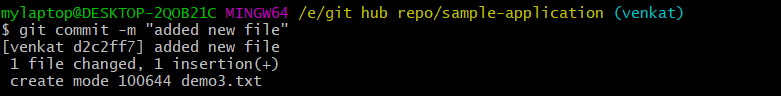


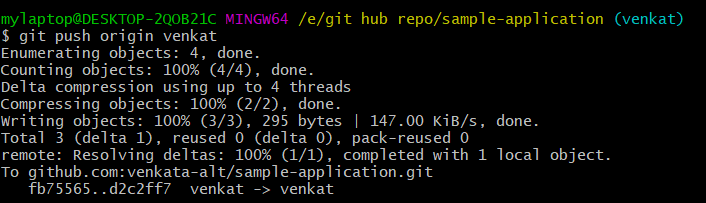
Open git bash create a file 🡺push to git hub🡺now we can see pull requests🡺merging is blocked🡺

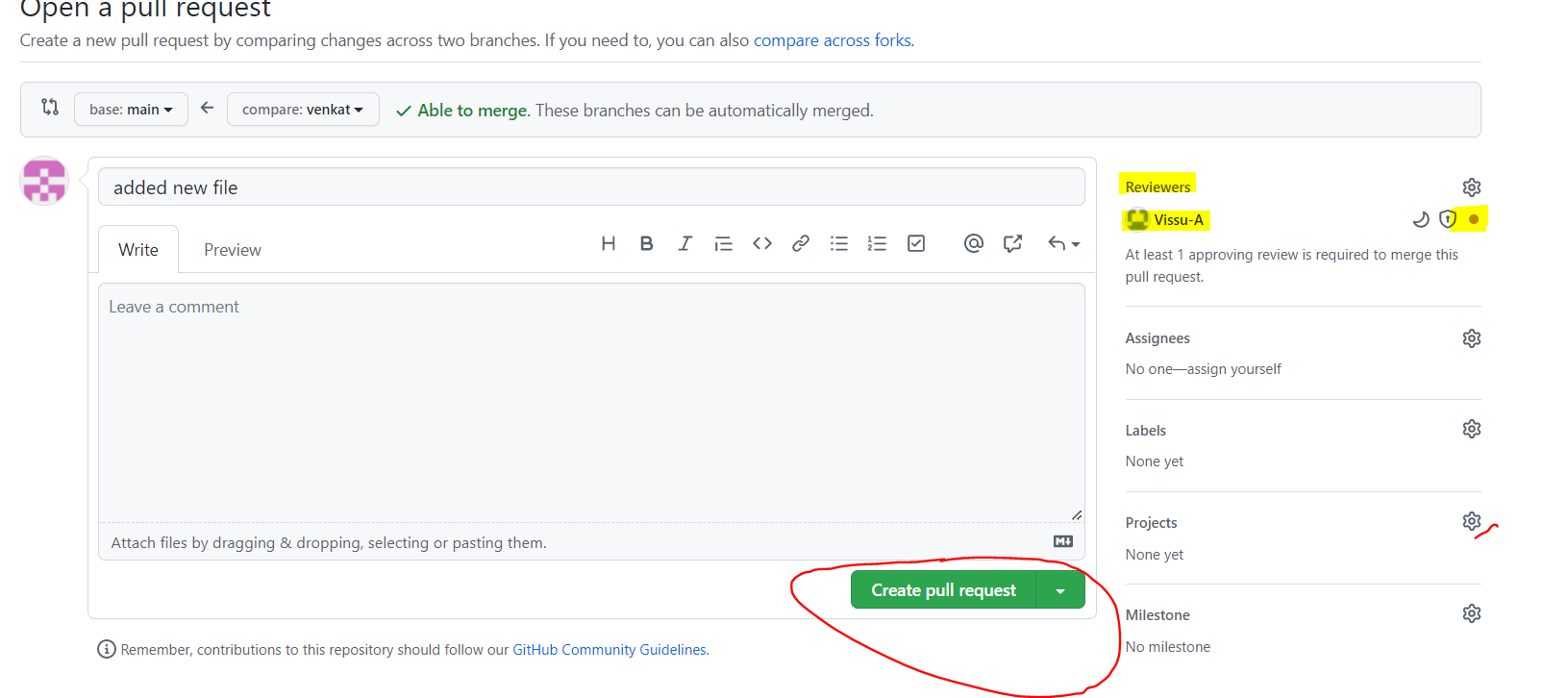
Ex:-echo “hi this is my file” > demo3.txt

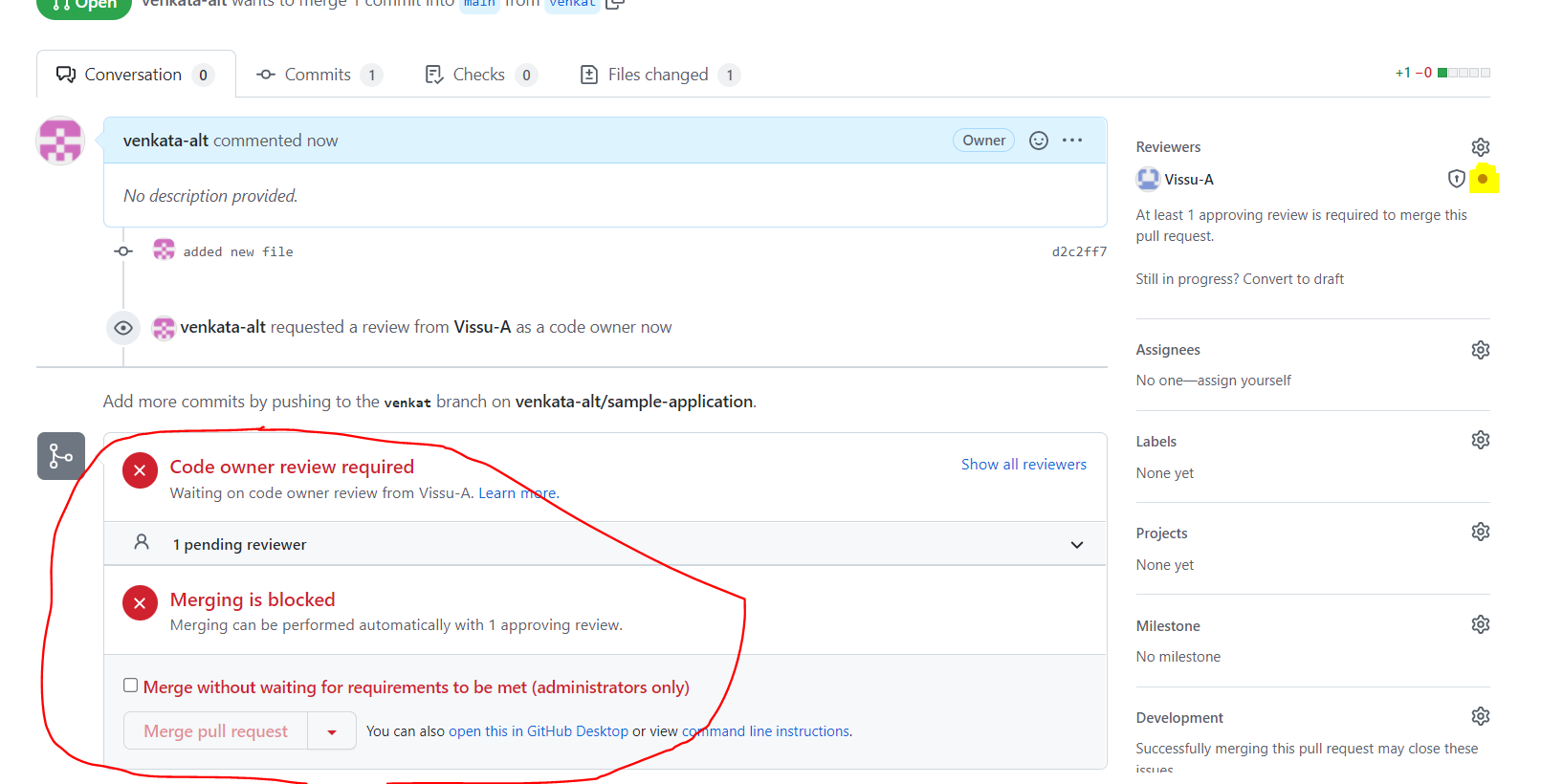




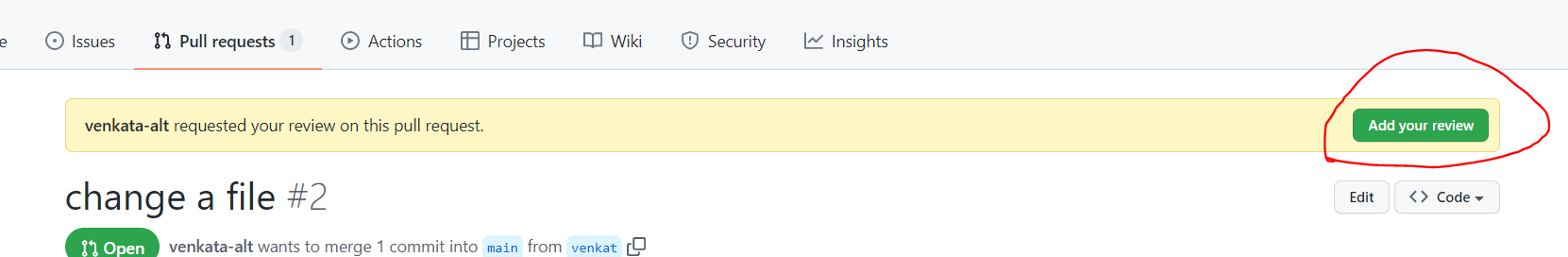




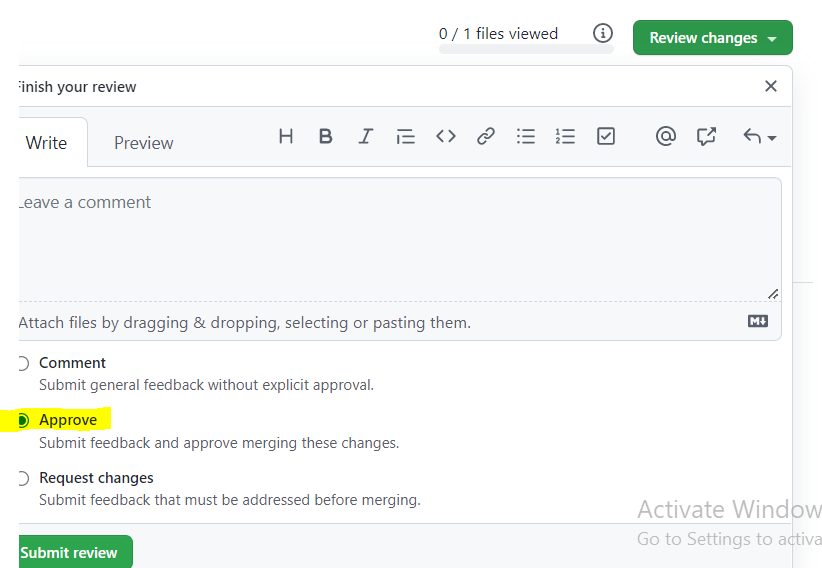




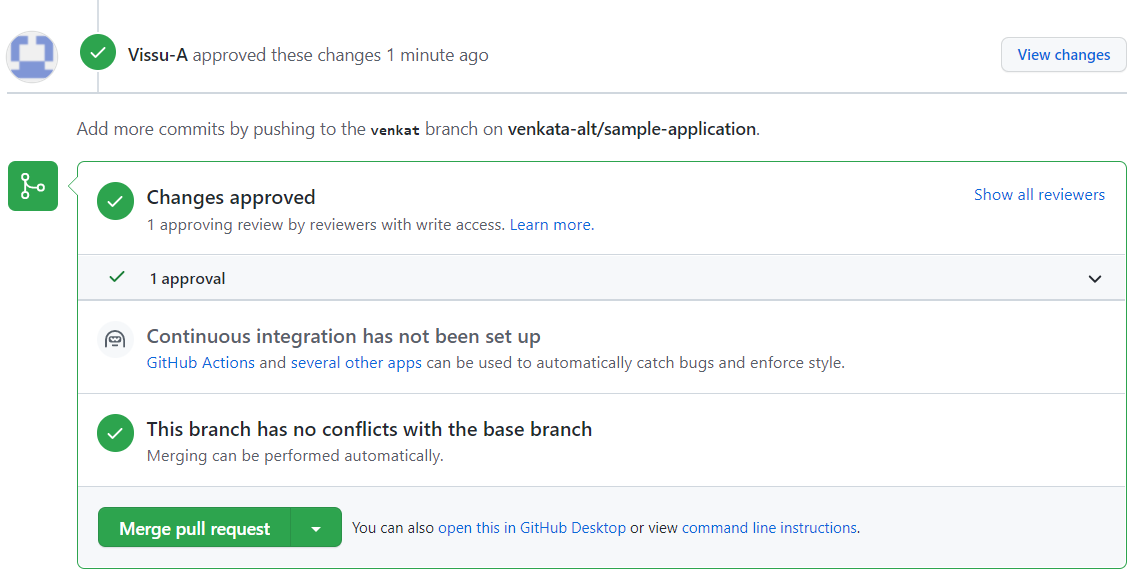
Why because collaborator don’t accept our request🡺 after click on collaborator notification buttion 🡺click on latest notification🡺 click on add your review



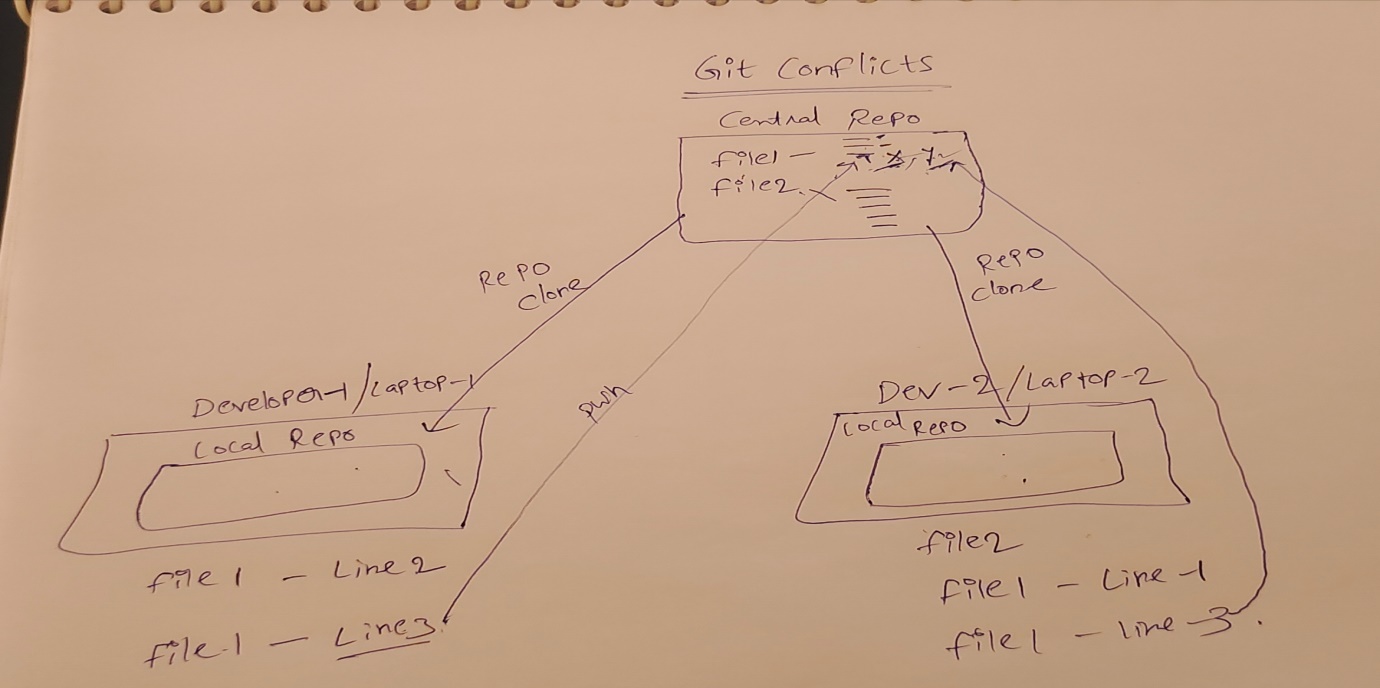
Review change🡺 approve🡺submit review



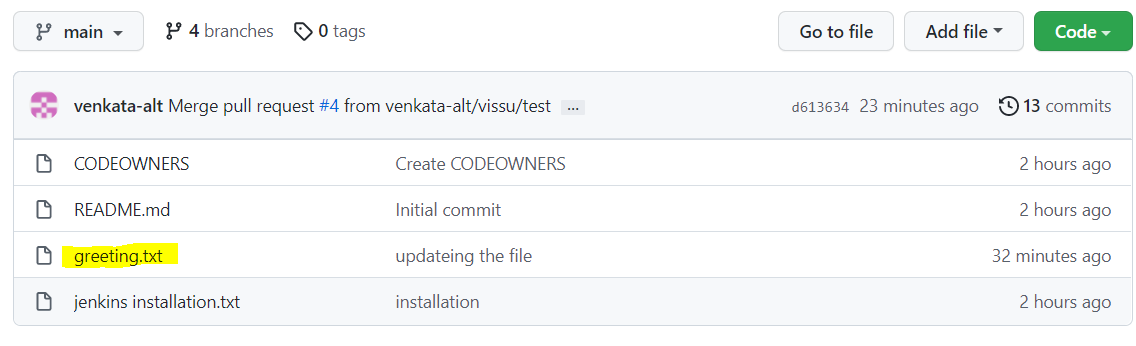
Now we can enable merging option on our repo🡺merge pull request🡺confirm merge



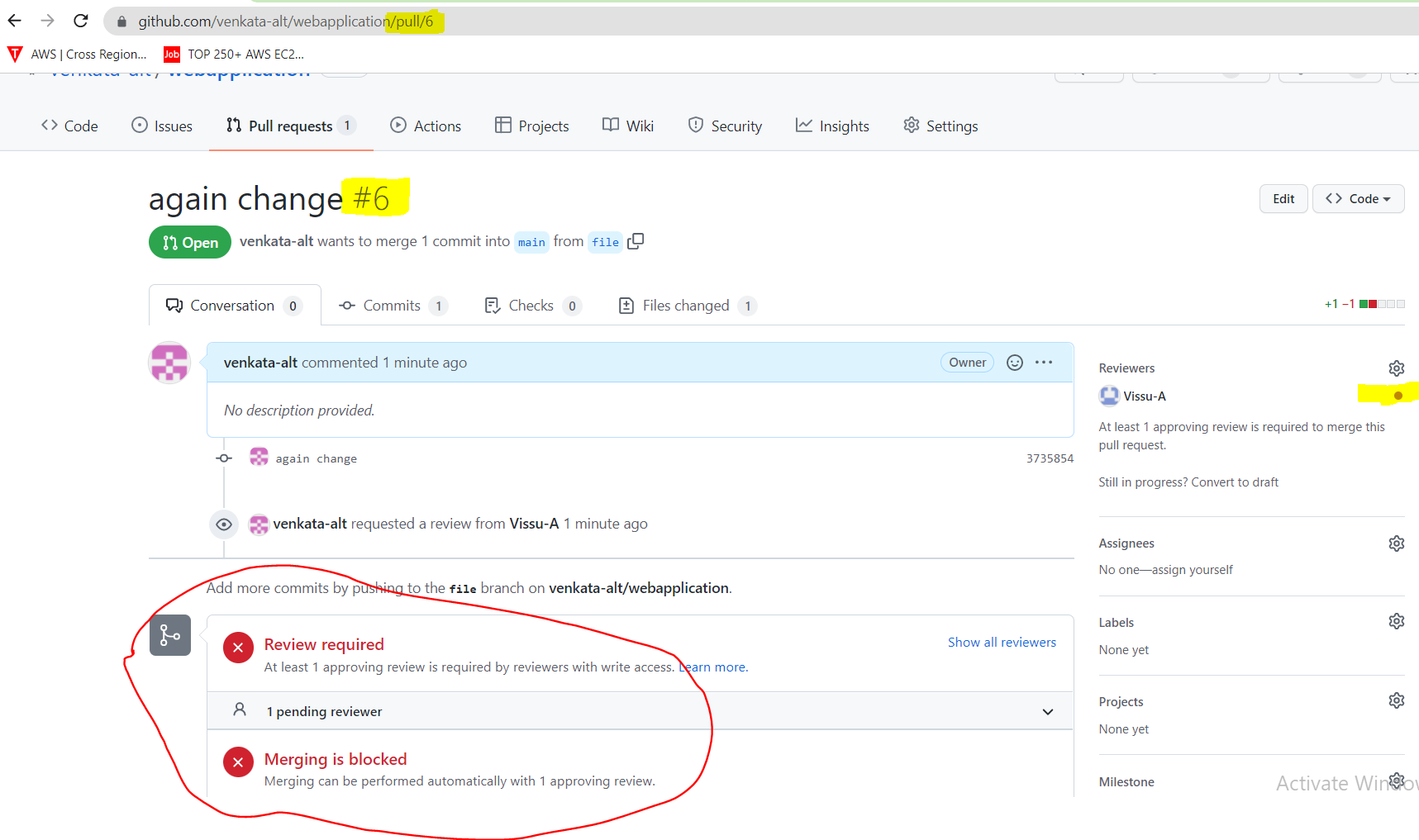
**Class 6:- GIT CONFLICTS**

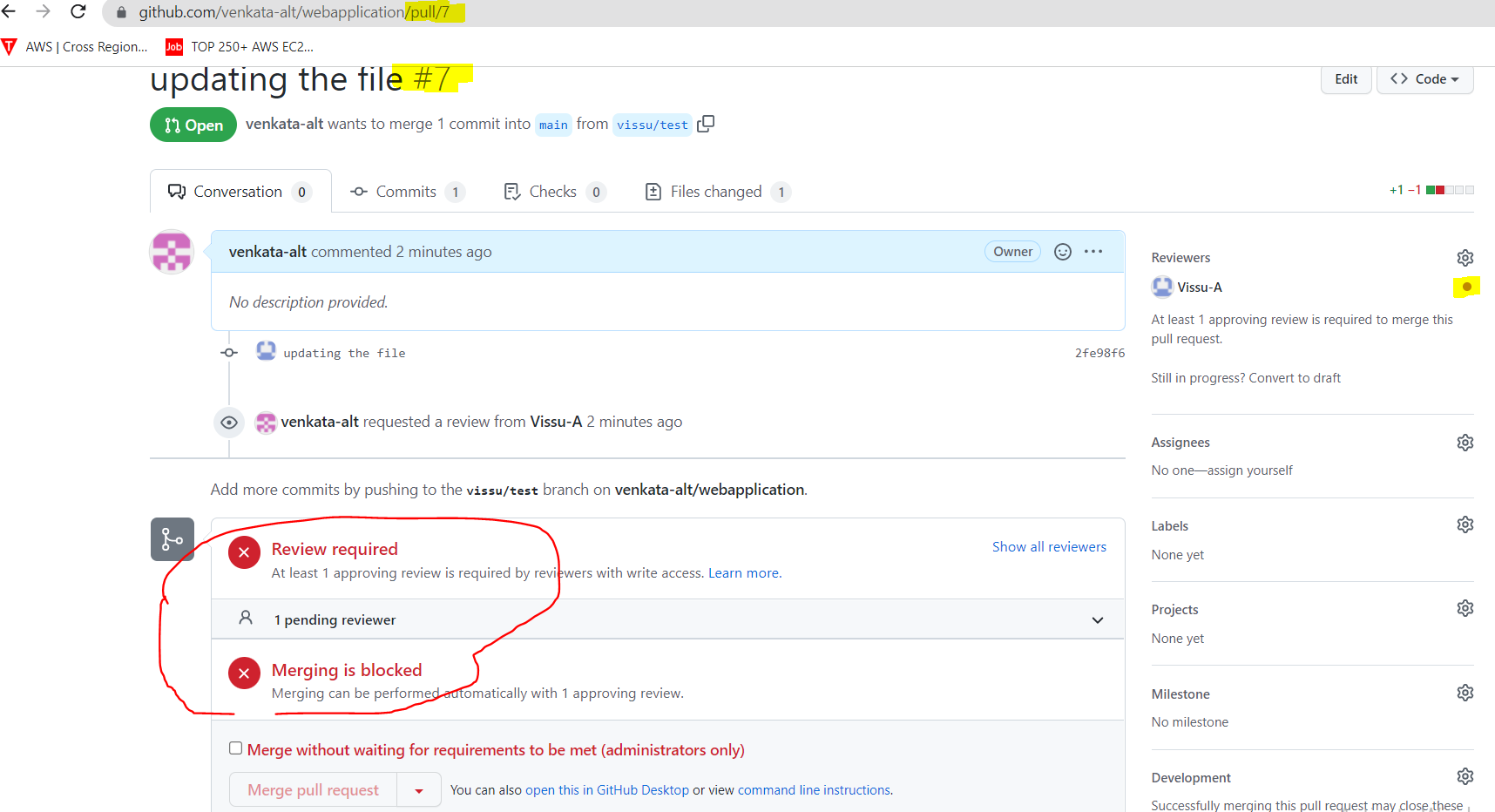
****

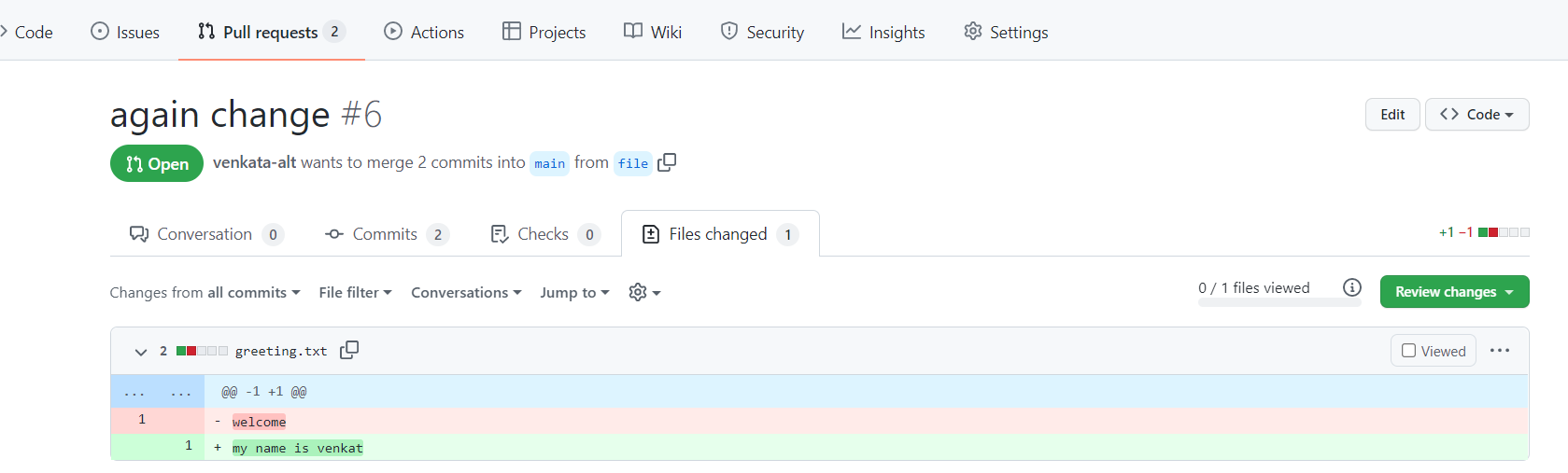
**Conflicts :-** Conflicts will come when two people are trying to modify the same line in the same file.



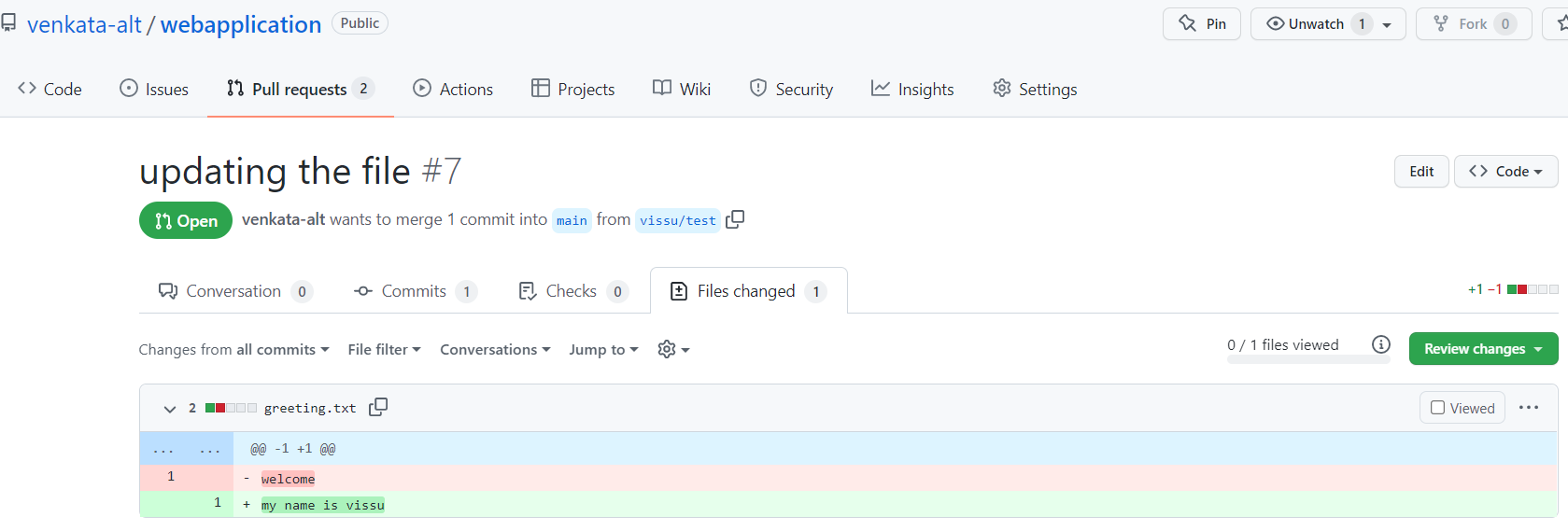
Now we will modify the greeting.txt file by two people, so we will create two pull request





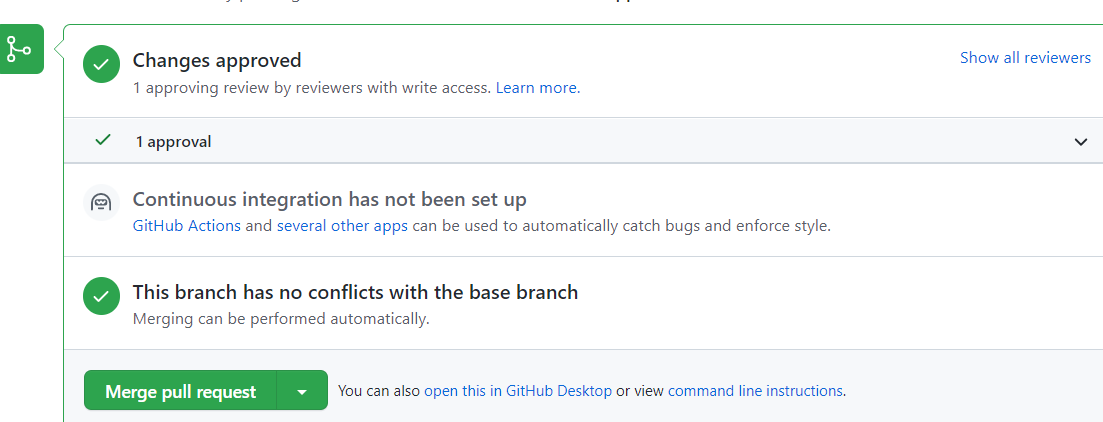


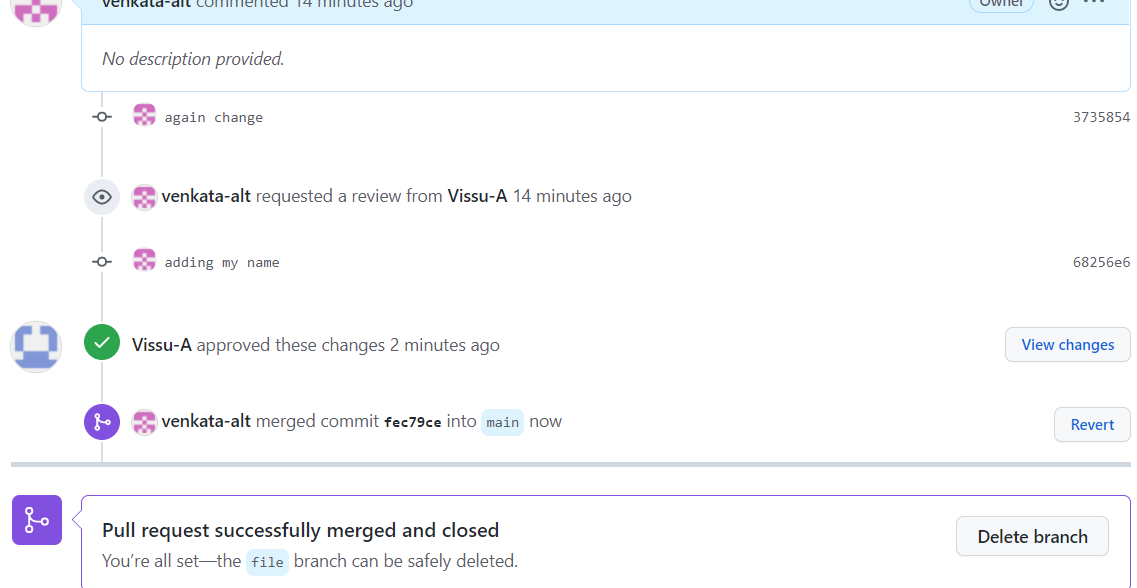
In pull request 6 we are modifying the line number 1 in the file, we removed welcome and added new line “my name is venkat”



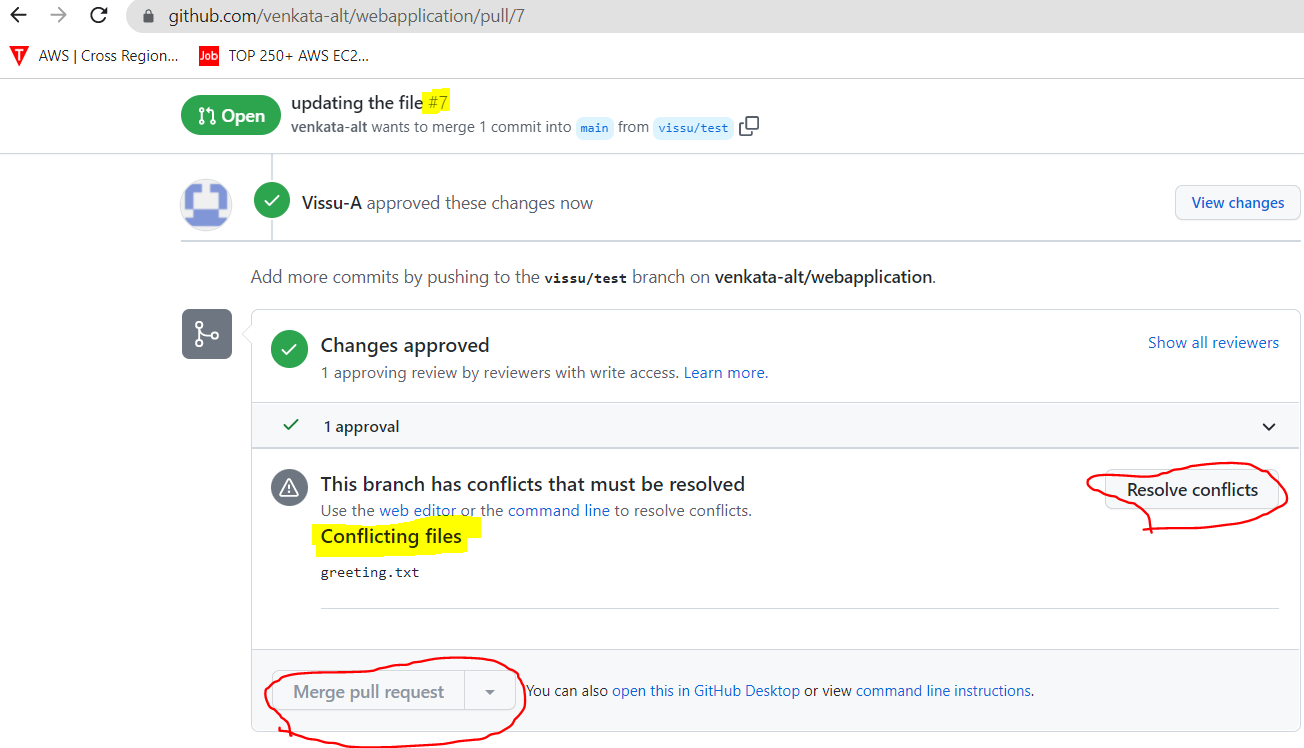
In pull request 7 we are modifying the line number 1 in the file, we removed welcome and added new line “my name is vissu”

Approved changes, merging the pull request

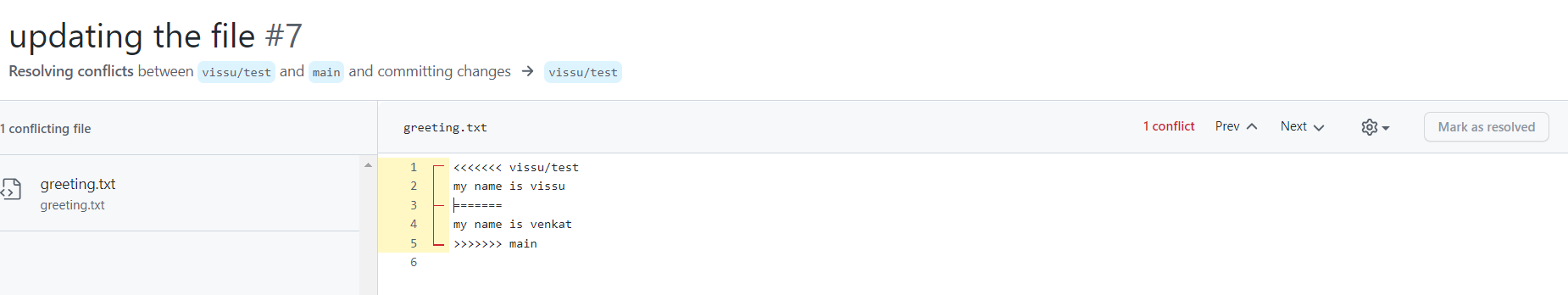




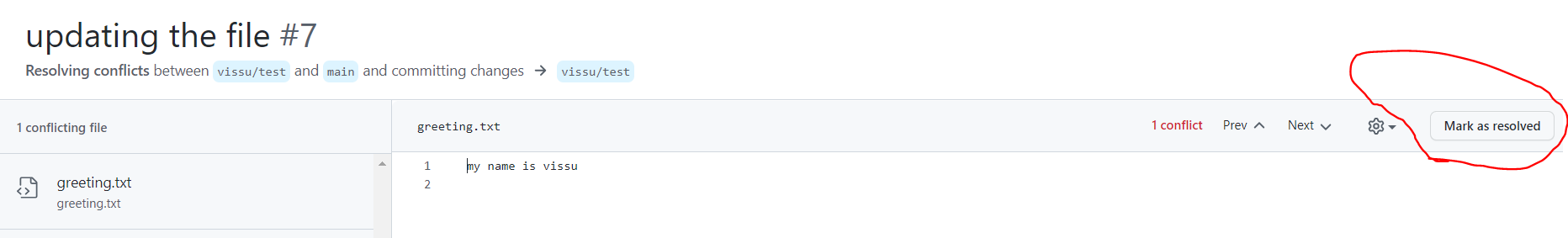
When we merge the pull request 6 changes immediately pull request 7 will have the conflicts

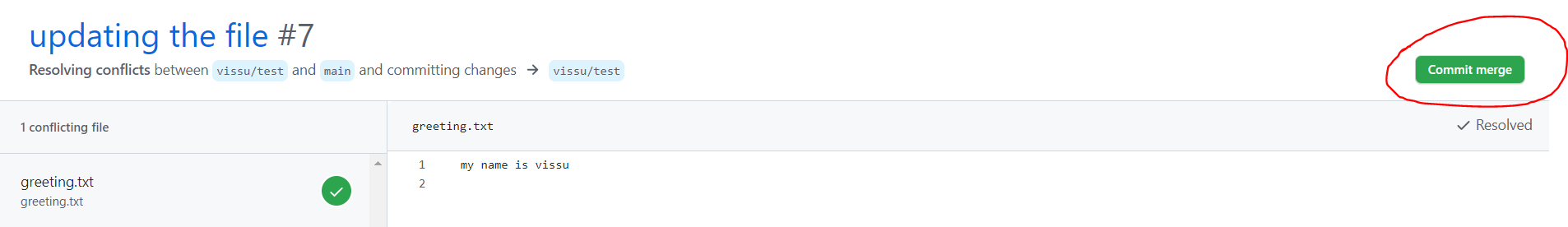


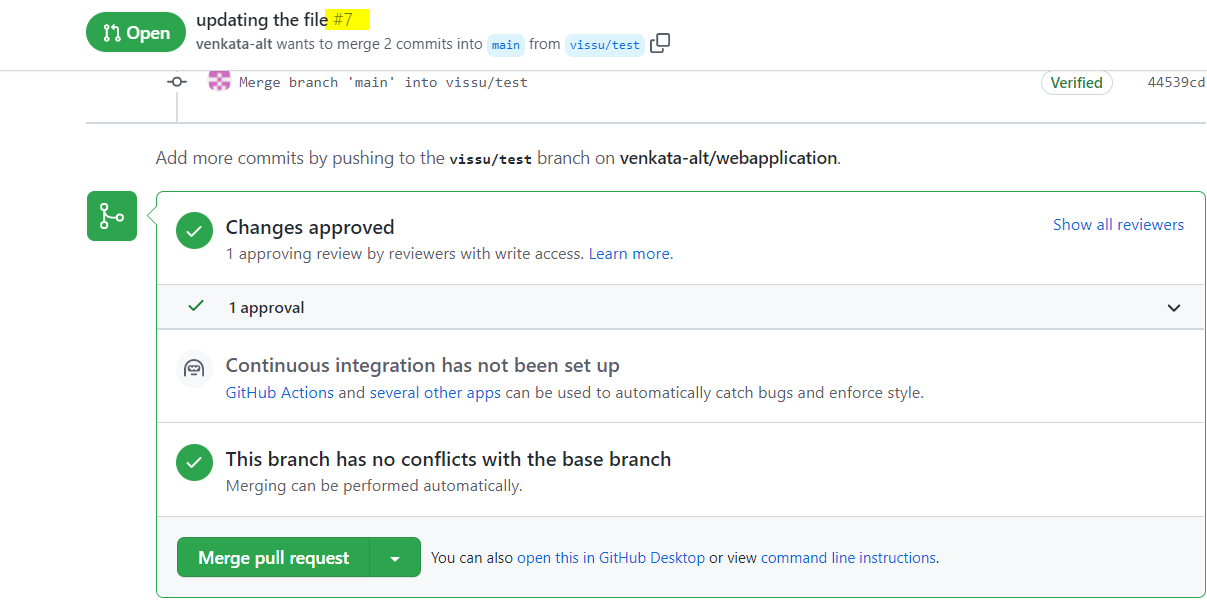
If we can see pull request 7 even the review is approved still merge pull request button is devolve, because of the conflict, so to resolve the conflict button



Now both people will have to connect to discuss what changes to be finalize and merge.

so once changes are finalized click on mark as resolved

click on commit merge

now merge pull request button is enabled, click on merge request for file 7



Click on confirm merge