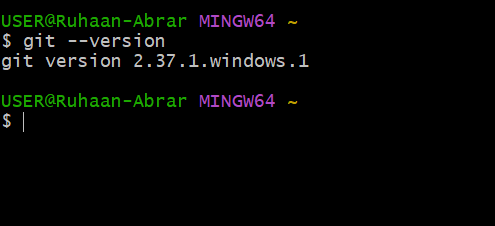
**Git Tasks**

1)Install git.



Installing Git is a straightforward process. Here's how you can do it:

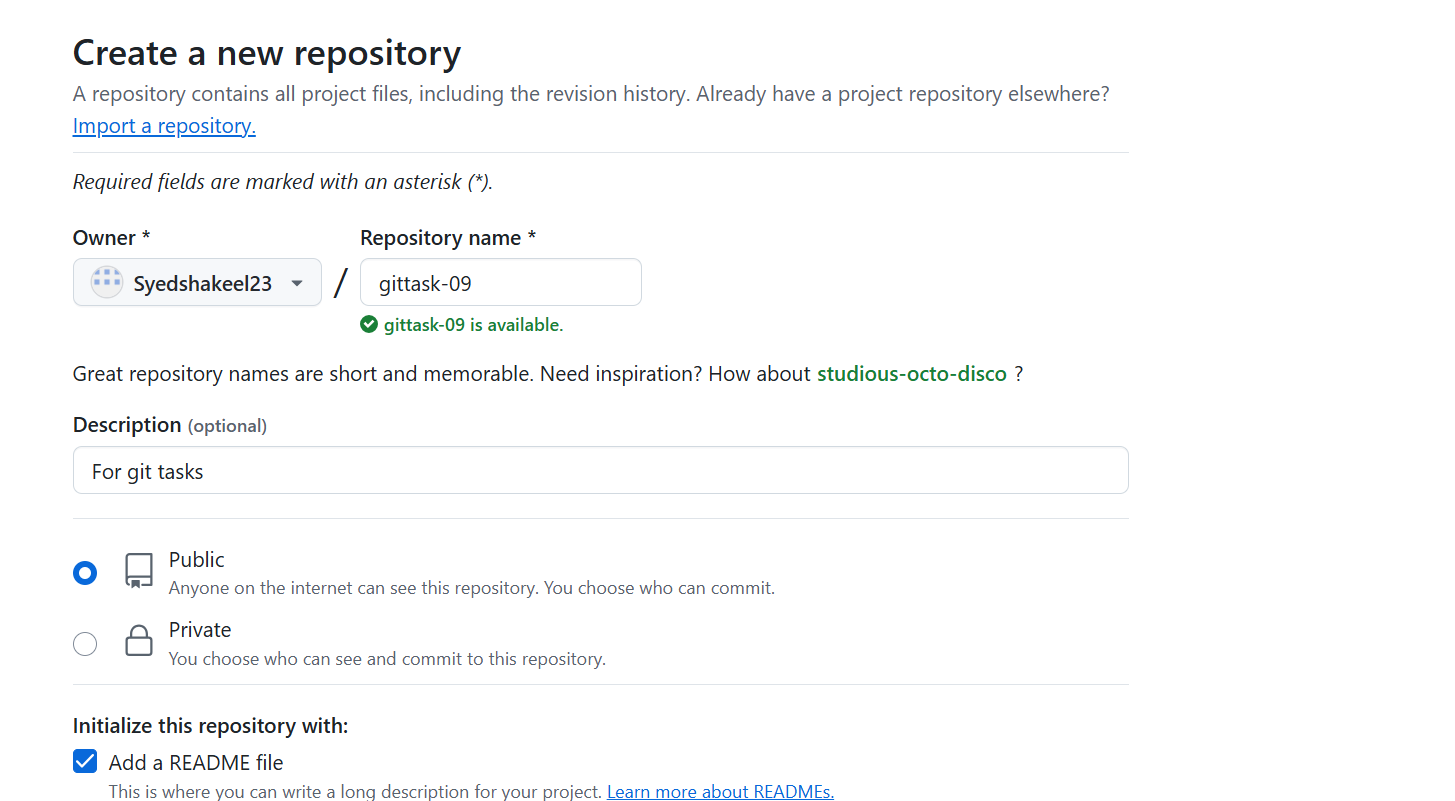
### **On Windows:**

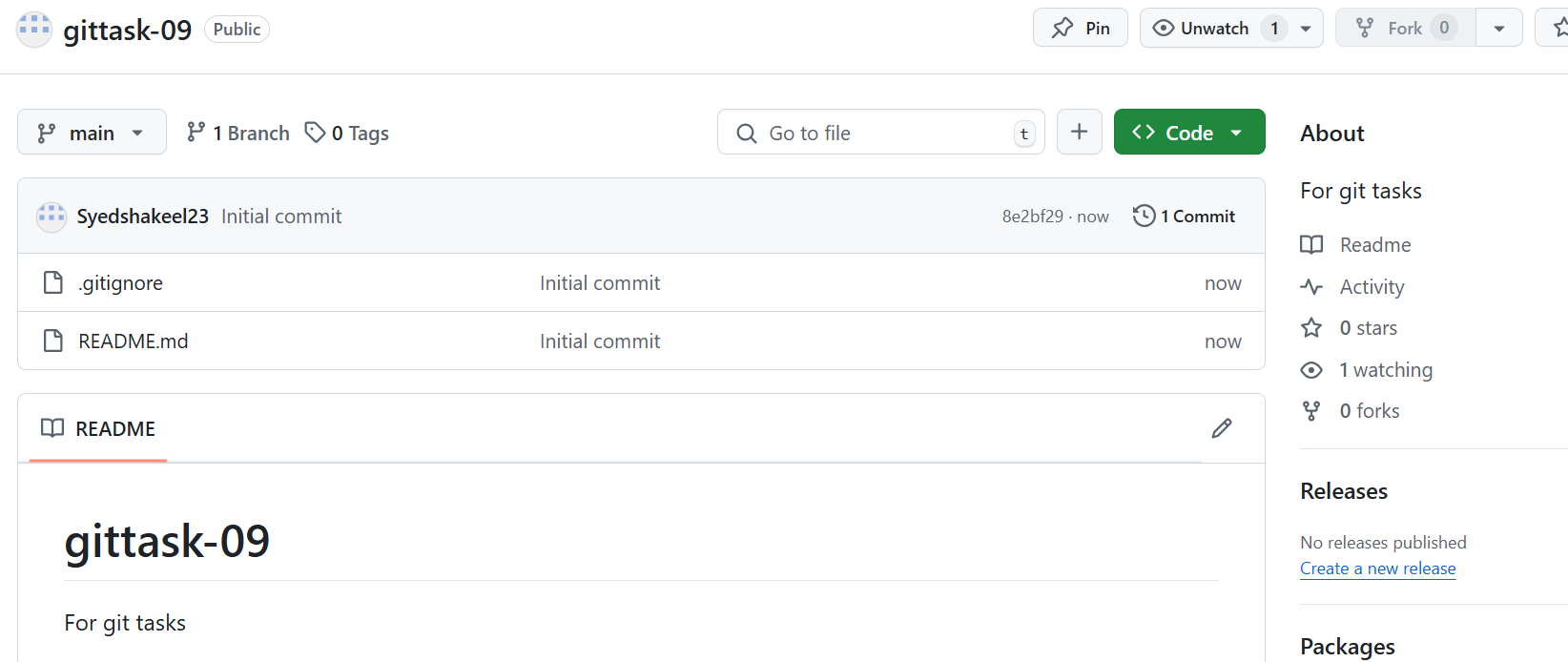
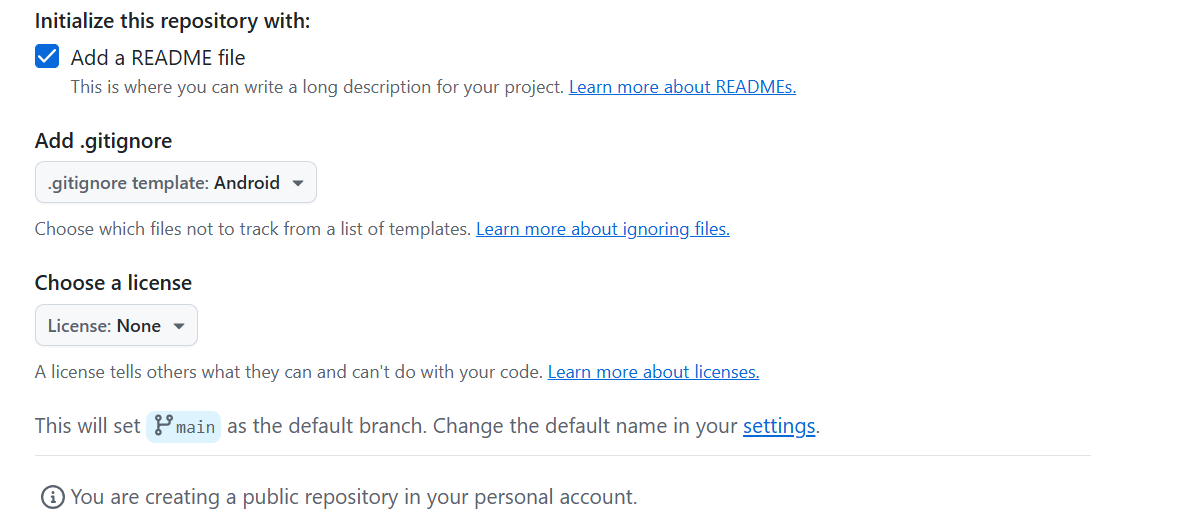
1. Download the latest Git installer from git-scm.com.
2. Run the installer and follow the installation steps.
3. Make sure to select options that suit your needs, such as adding Git to your system PATH.
4. Once installed, open a command prompt and type git --version to verify the installation.

2)Create a repo in github with README.md and .ignore file.

### **Using GitHub Website:**

1. Go to GitHub and log in.
2. Click on the **"+"** icon in the top-right corner and select **New repository**.
3. Enter a name for your repository and add a description if needed.
4. Choose **Public** or **Private** depending on your preference.
5. Check the box **"Initialize this repository with a README"**.
6. Select the .gitignore **template** based on the type of project (e.g., Python, Node.js).
7. Click **Create repository**.





3)Clone the created repo to local.

Open Gitbash

Navigate to the directory where we want to clone the directory

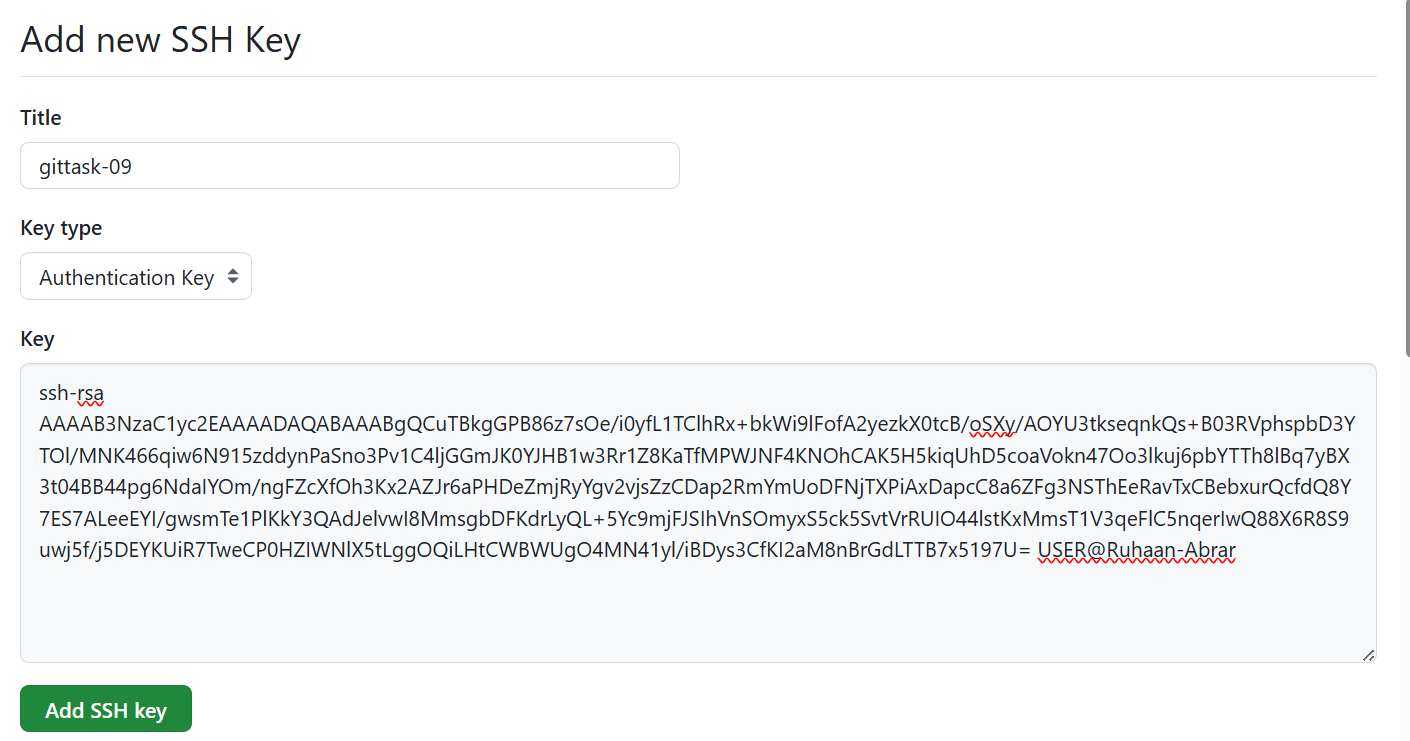
cd <Directory>

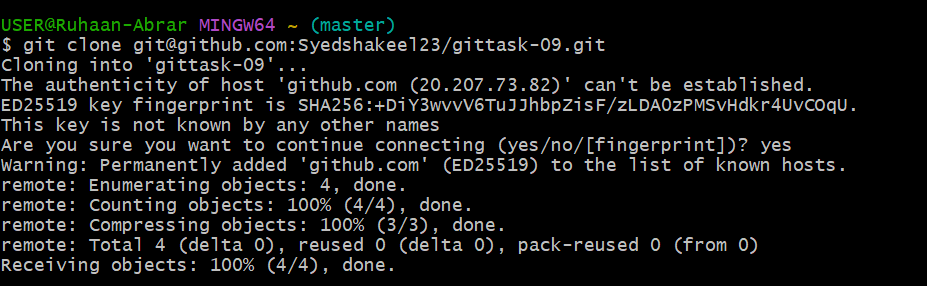
Run git clone command with repository URL git@github.com:Syedshakeel23/gittask-09.git

To establish the connection with ssh we need to create ssh-keygen and copy that public key to repository

Once cloning is completed navigate into the cloned repository

We can verify the repository status using git status





4)Create two files in local repo.

Navigate to the local repository folder

Create two new files using the touch command



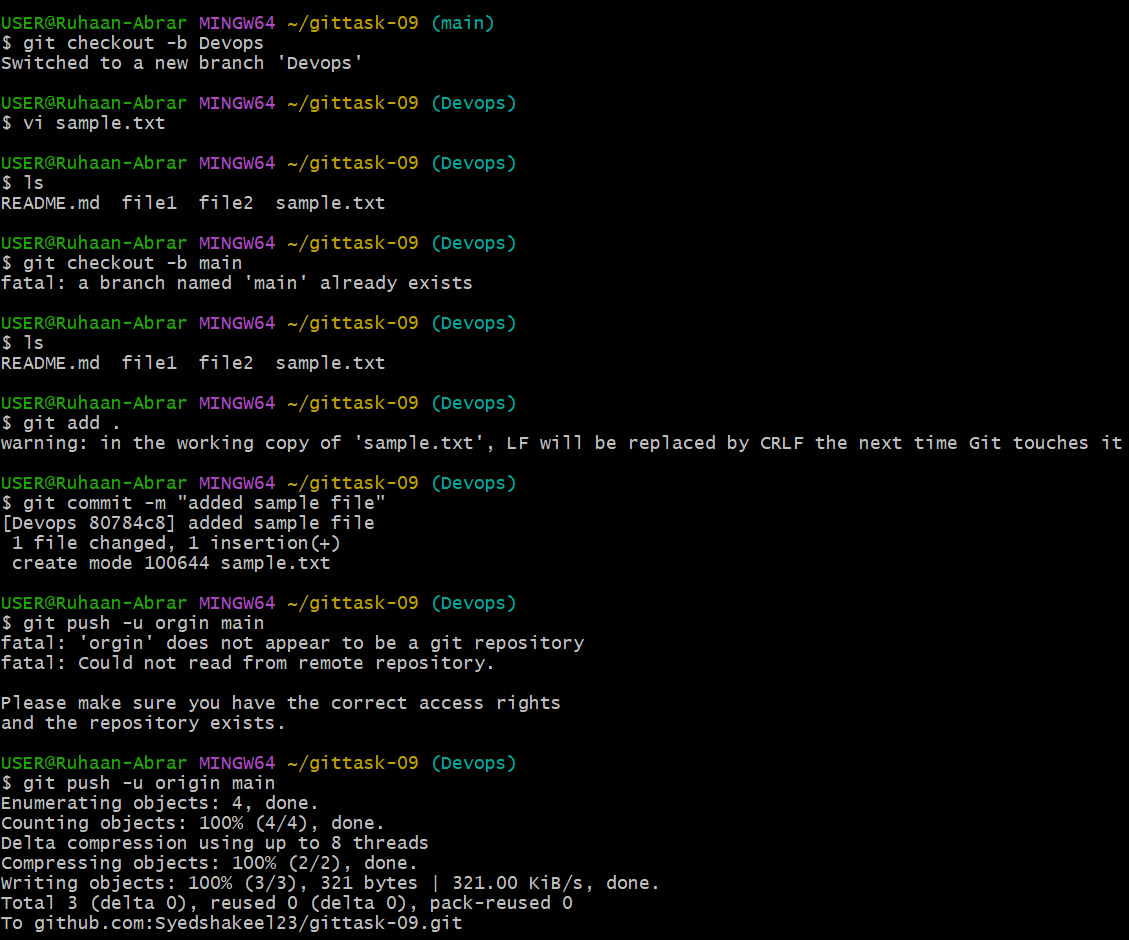
5)Commit two files and push to central Repository.

Add the files to Git tracking:

git add file1.txt file2.txt

git commit -m "Added file1.txt and file2.txt"

Push the changes to the remote repository (GitHub): git push origin main



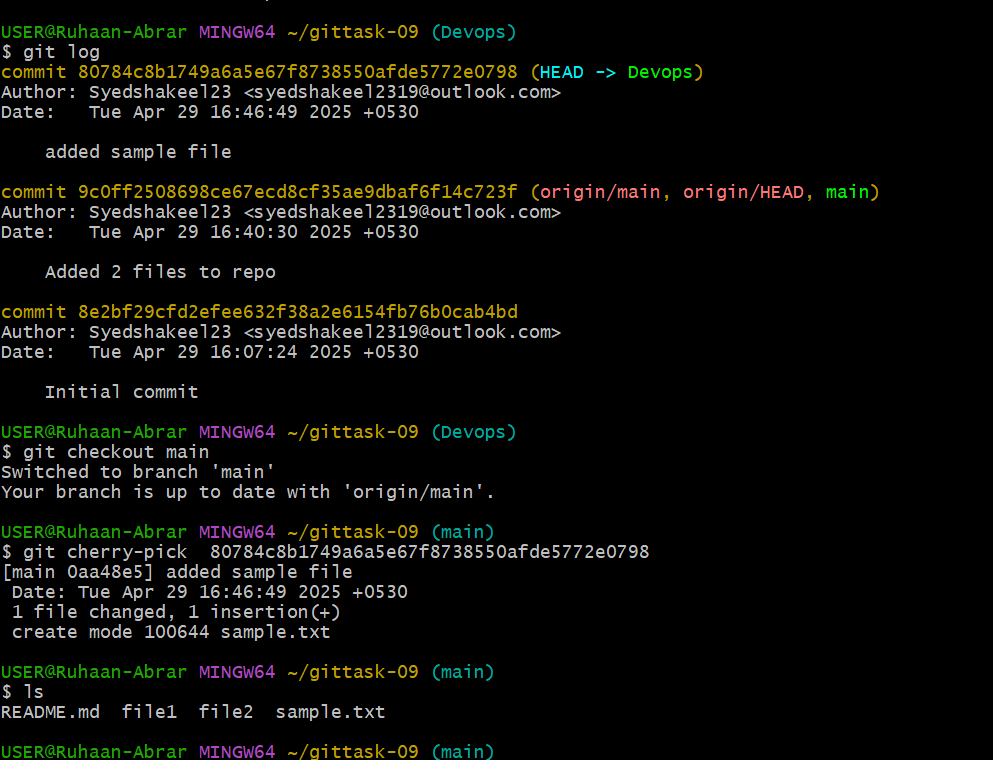
6)Create a branch in local and create a sample file and push to central.

Create a new branch using : git branch Devops or git checkout-b Devops

Create a sample file: touch sample.txt

Add and commit the file : git add sample.txt and git commit -m "adding the sample file"

Push the file to central repo : git push -u origin main



7)Create a branch in github and clone that to local.

### **1. Create a Branch on GitHub**

1. Go to GitHub and open your repository.
2. Click on the **Branches** dropdown (near the top left).
3. Enter a name for your new branch (e.g., new-feature-branch) and click **Create branch**.

Clone the created repo to local.

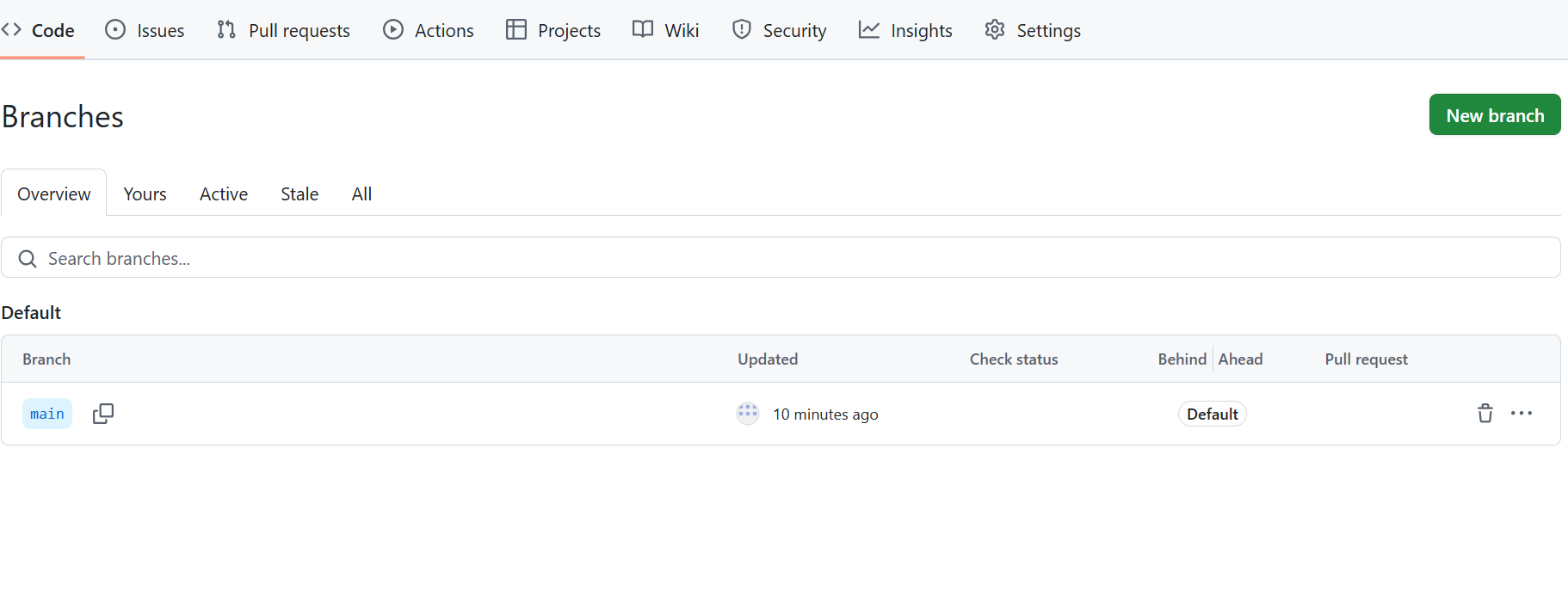
Open Gitbash

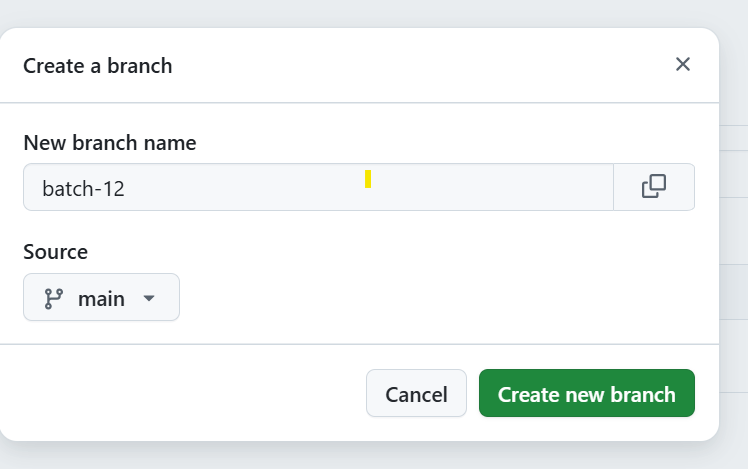
Navigate to the directory where we want to clone the directory

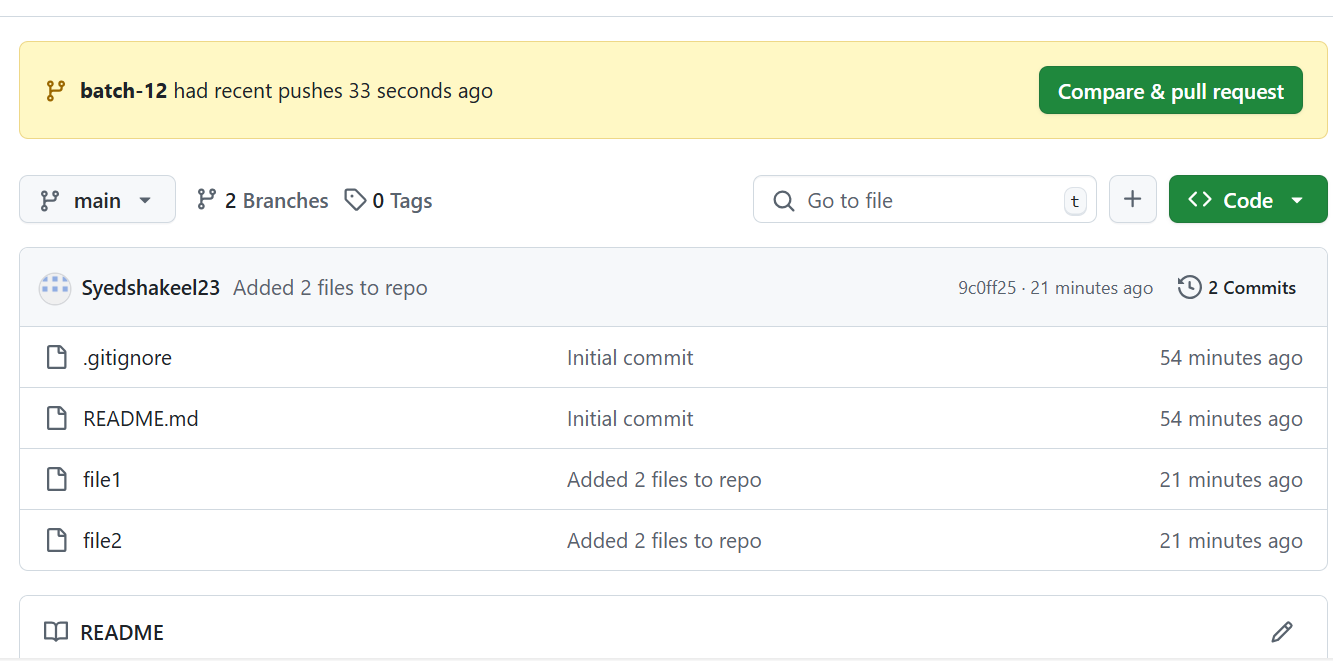
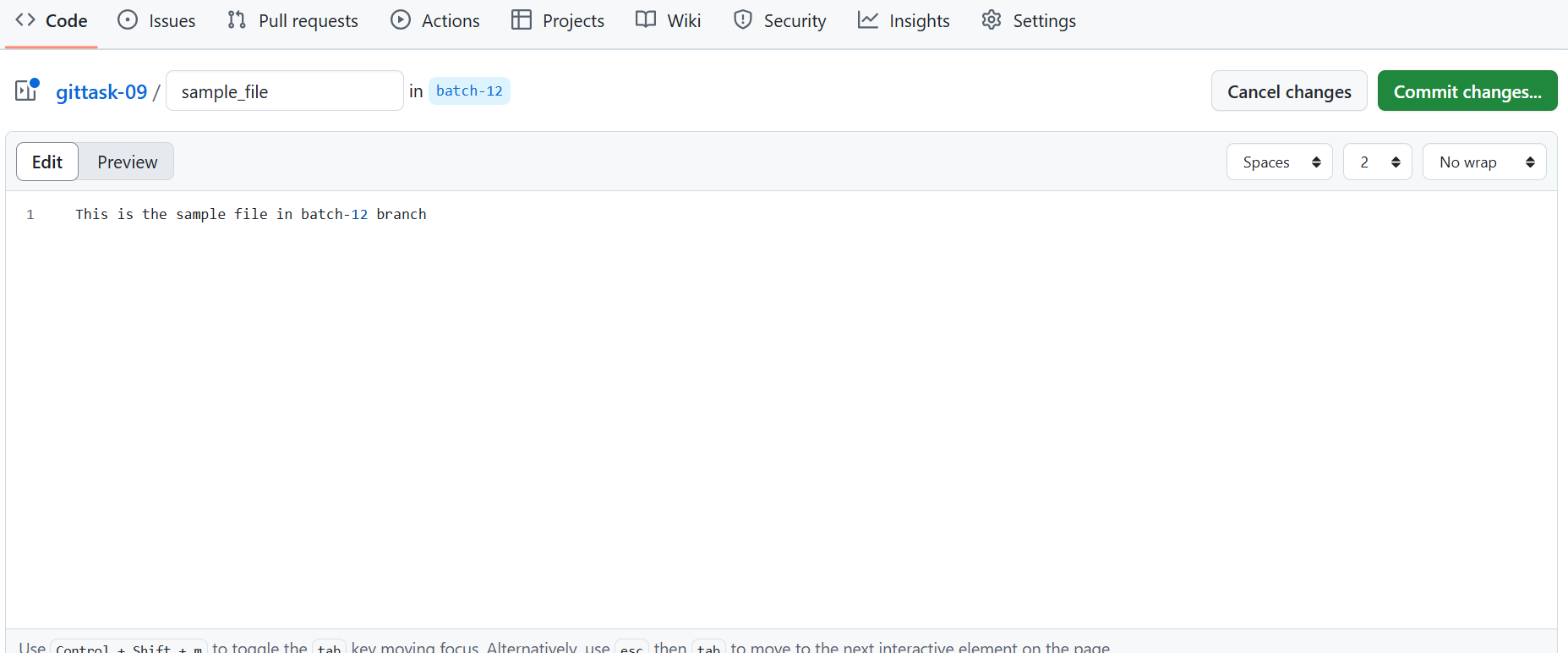
cd <Directory>

Clone the Repository : git clone git@github.com:Syedshakeel23/gittask-09.git

Switch to the new branch created : git fetch origin and git checkout batch-12







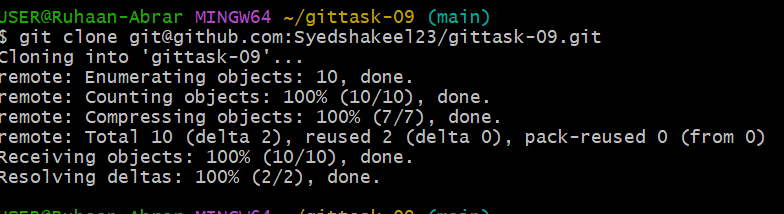
8)Merge the created branch with master in git local.

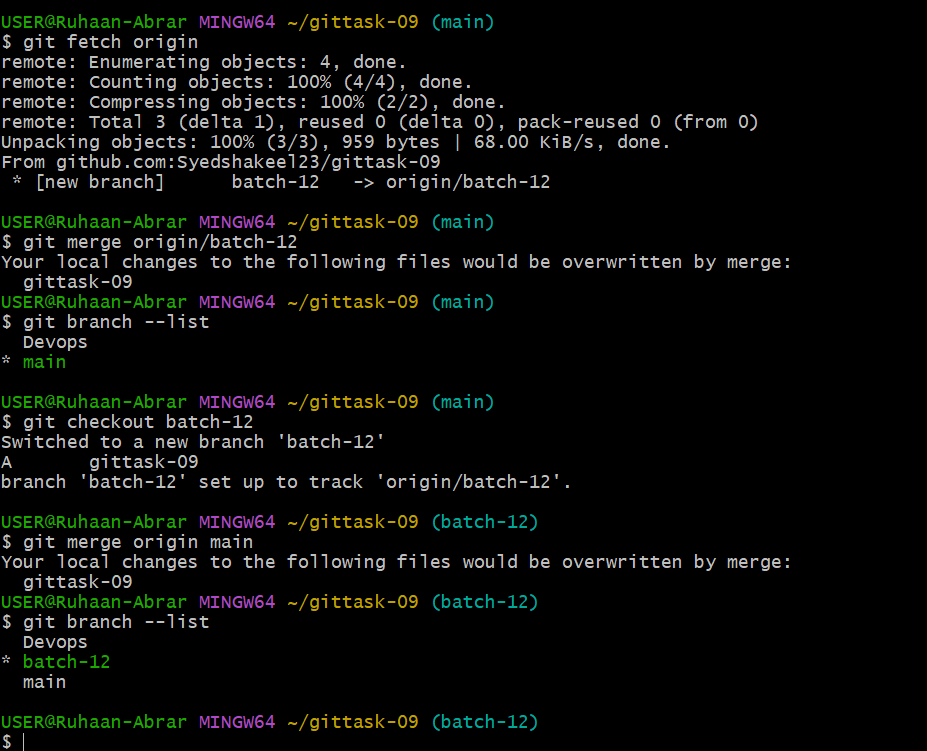
Switch to the main branch : git checkout main

Merge the branch : git merge batch-12

Resolve and merge conflicts manually : git add . And git commit -m “merge conflict resolved”

Push the changes to central repo : git push –u origin main





9)Merge the created branch with master in github by sending a pull request.

Push the branch to Central repo if not pushed already

git push -u origin batch-12

Open a Pull request on GitHub

Go to GitHub and open repository

Click on the "Pull requests"

click "New Pull request"

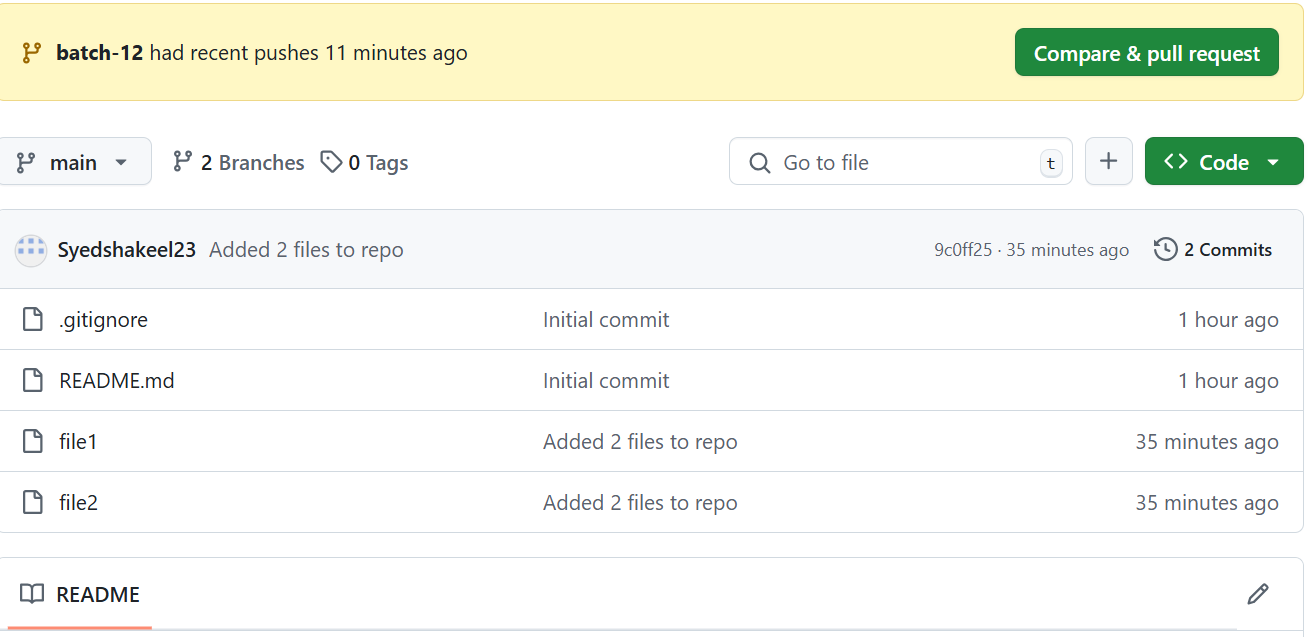
Select our branch batch-12 in the compare dropdown

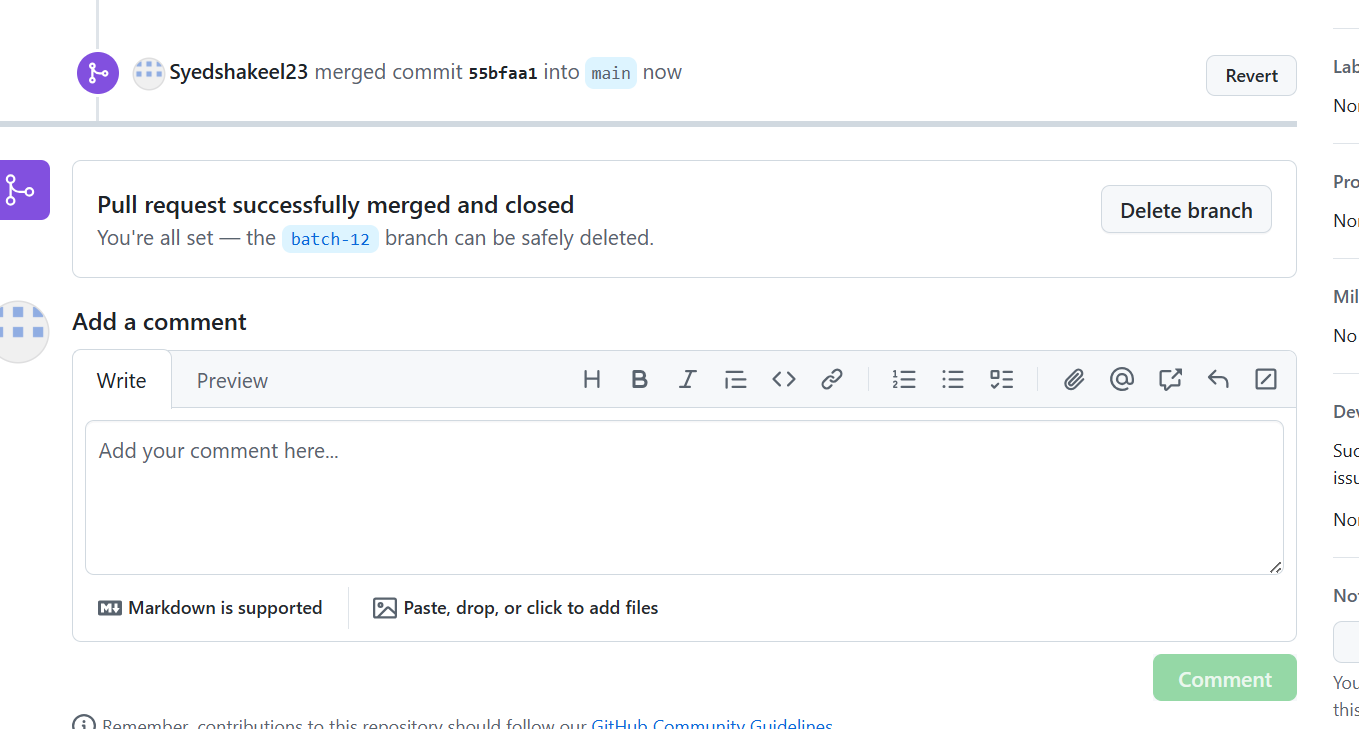
Ensure the base branch is main

Review the changes and check the details and click "Create Pull Request"

Review and Merge the Pull request

Once the request is approved click Merge pull request and confirm merge.





10)create a file in local and send that to branch in github.

Navigate to the local repository cd /gittask-09/

Switch to the branch main : git checkout main

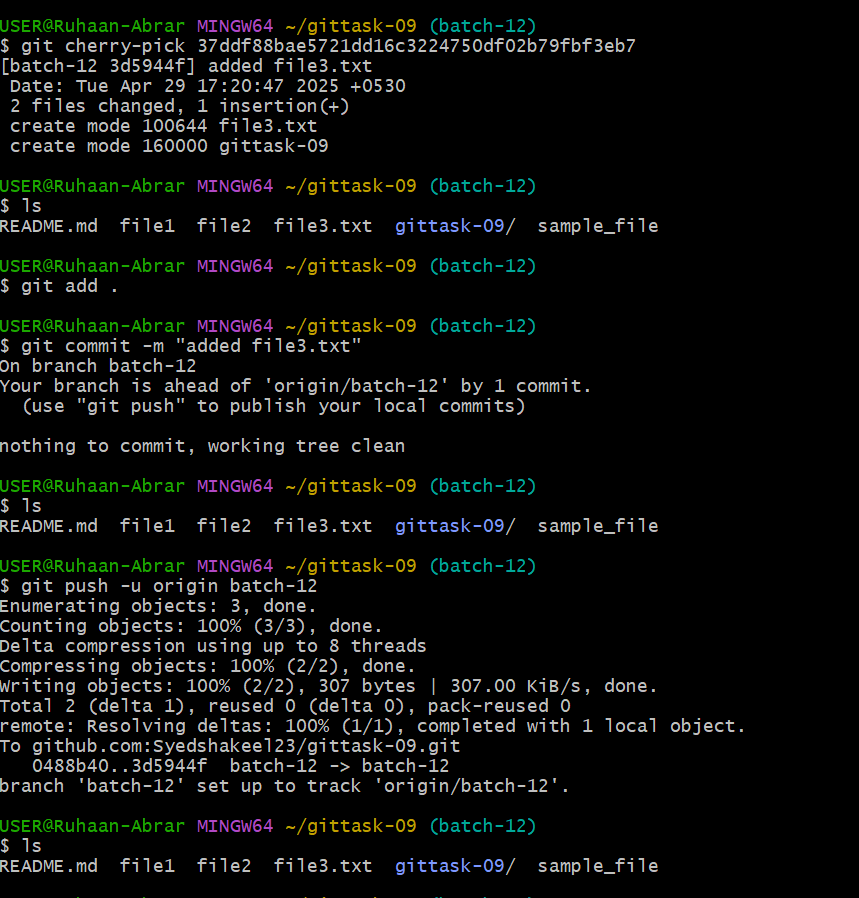
create a new file touch file.txt

Add and commit to the file

git add file.txt

git commit -m "Added new file to main"

Push it to GitHub : git push -u origin main



11)clone only a branch from github to local

Navigate to directory cd gittask-09

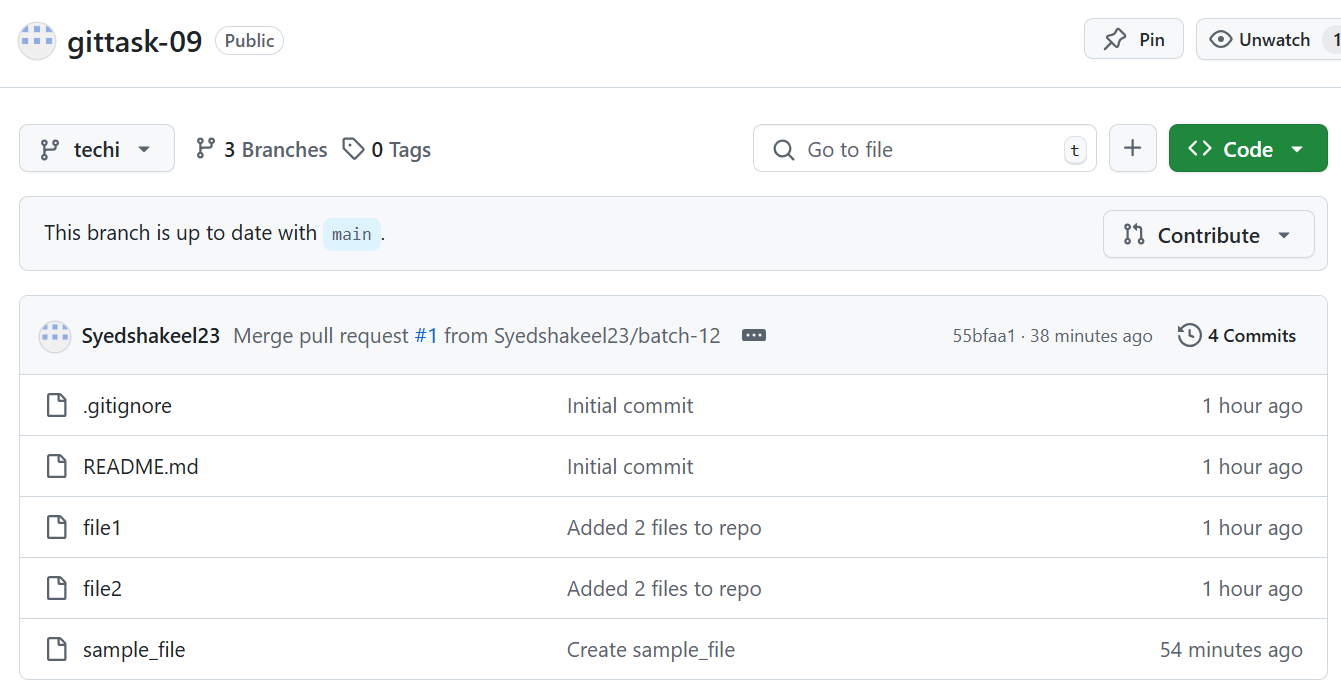
clone only specific branch :

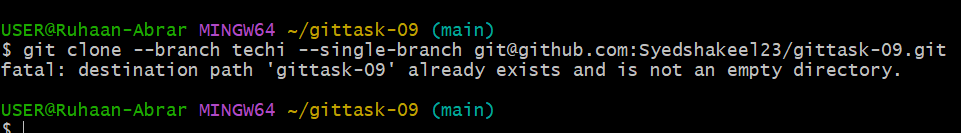
git clone --branch batch-12

Navigate into the cloned repository

cd gittask-09

check the branch details and status : git branch git status





12) create a file with all passwords and make that untrackable with git.

Create file touch passwords

add the file to .gitignore

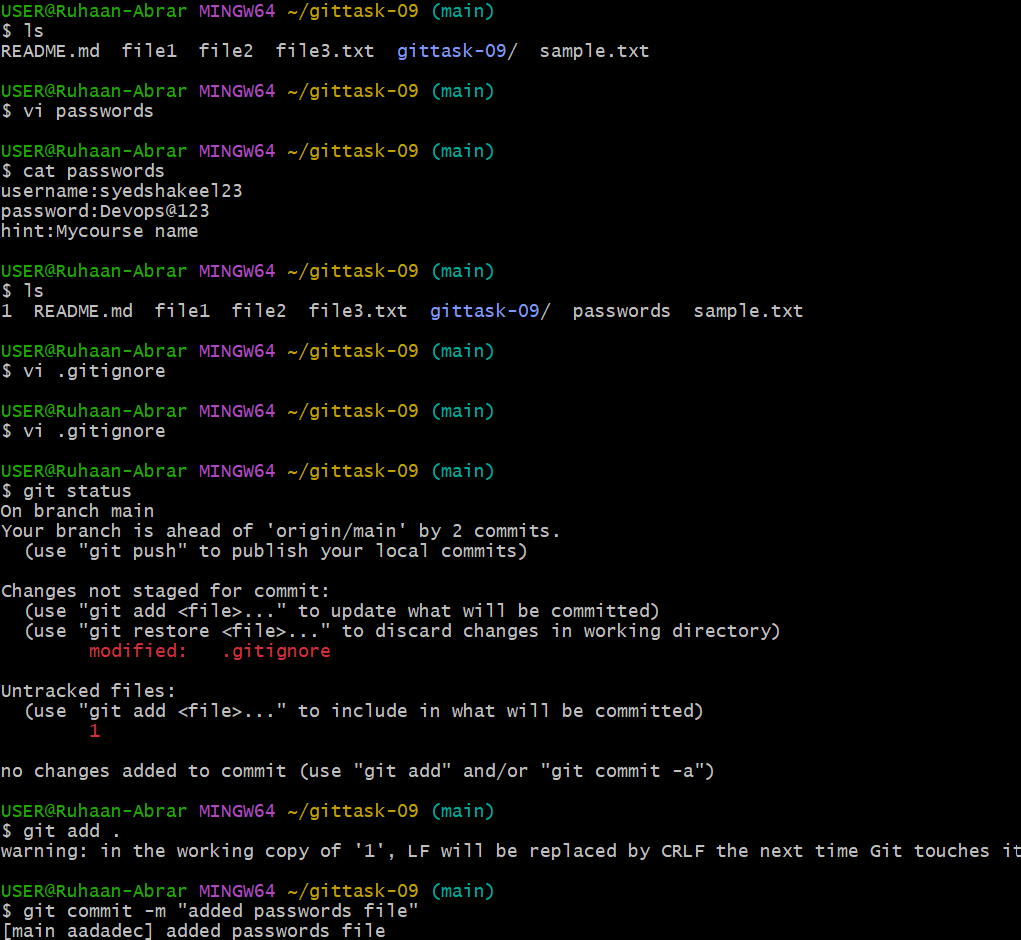
vi .gitignore

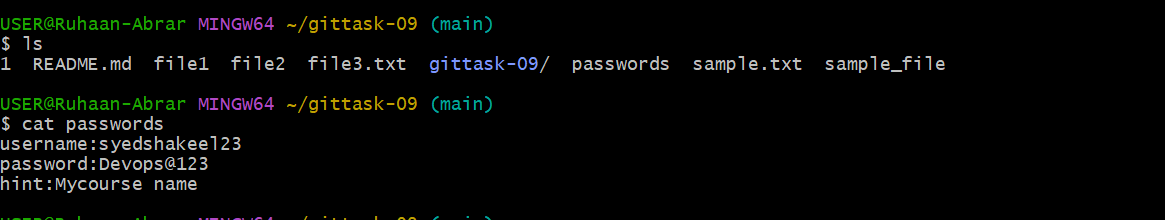
passwords

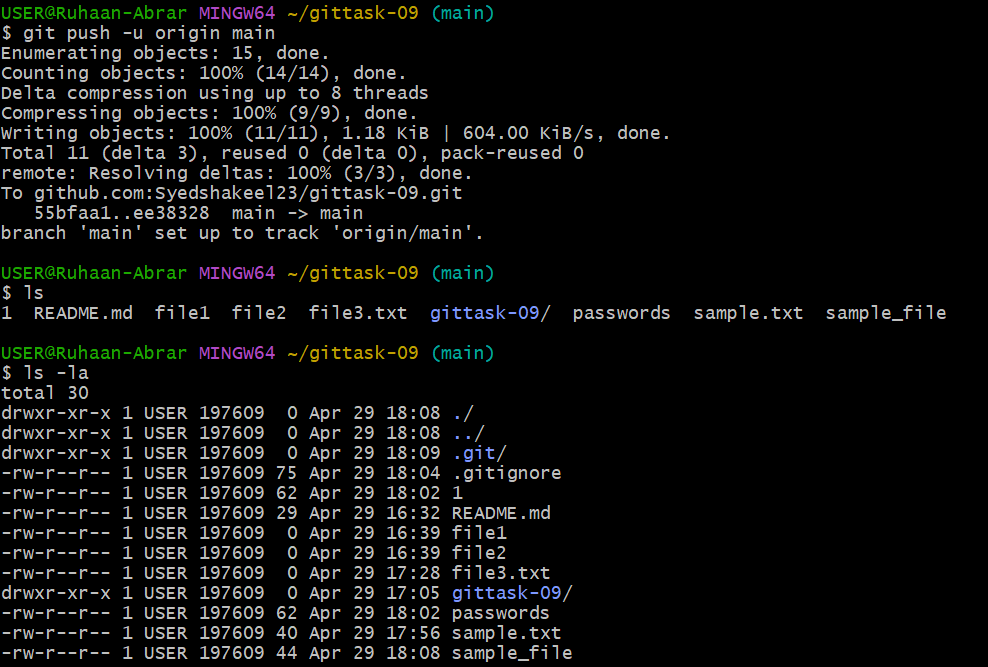
git add .gitignore

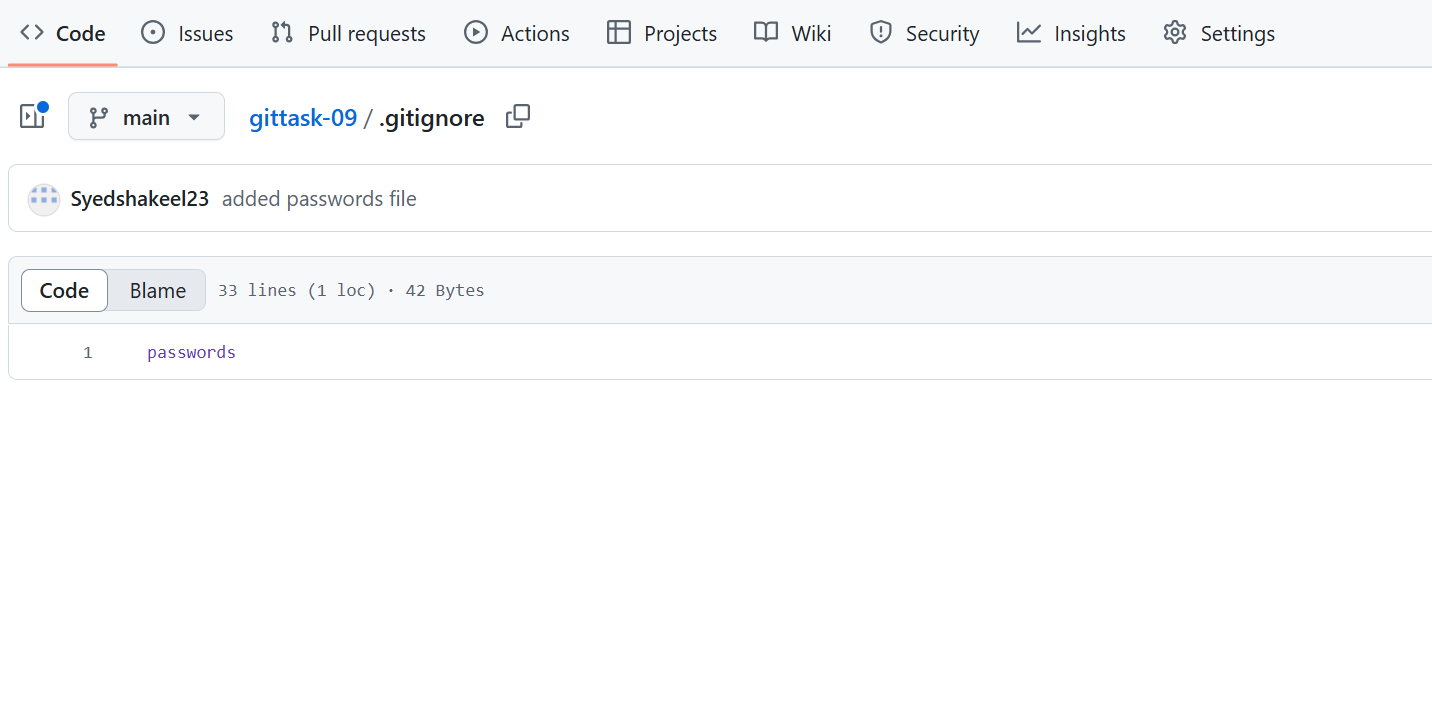
git commit -m "Updated .gitignore with passwords file"

Go to GitHub and check the file details we will be able to see password file but whats there in it we will not be able to see.









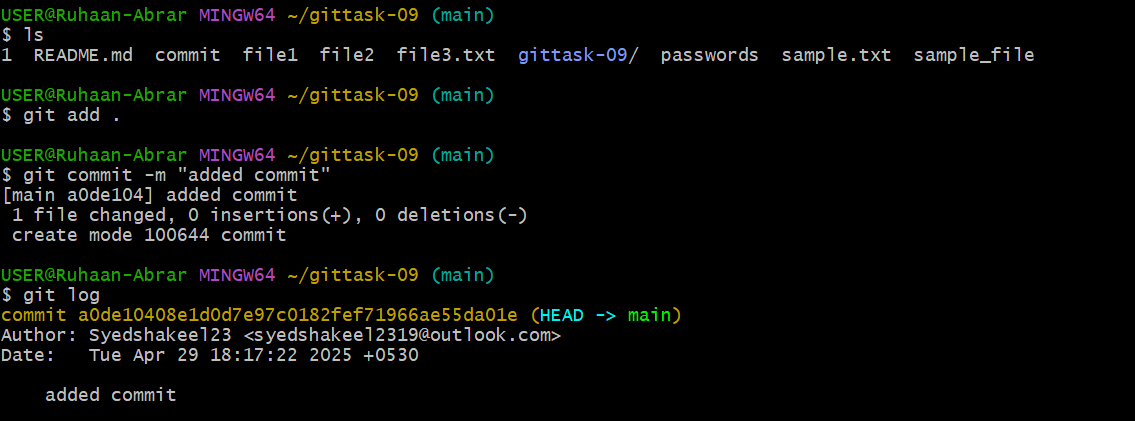
13)make a commit and make that commit reset without savings changes.

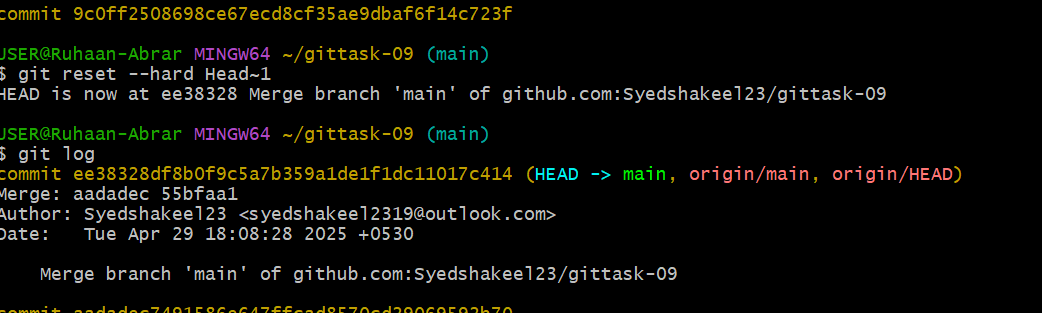
git add .

git commit -m "commit reset"

Reset the commit and discard the changes using

git reset --hard HEAD~1





14)Revert a commited commit to the older version.

git log

copy the commit id

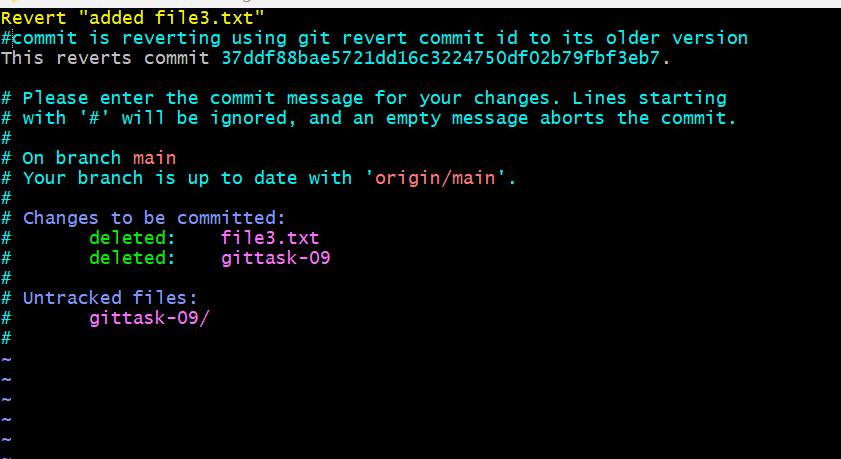
git revert commit id

git log

Reset to older commit

git reset --hard commit id

git push



15)push a file to stash without savings the changes and work on another file.

git status

git stash stash.txt

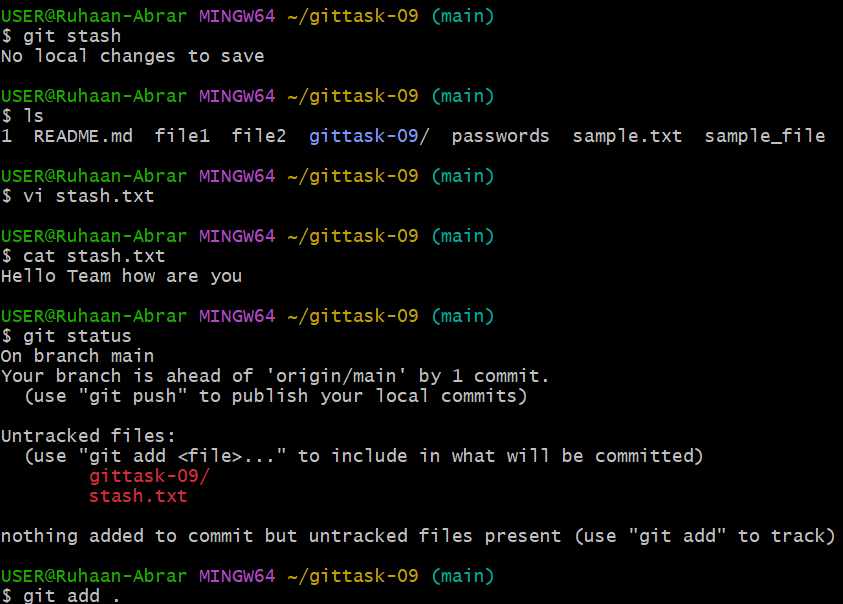
Continue working on another file

touch newfile.txt

vi newfile.txt

git add newfile.txt

git commit -m "Working on another newfile"





16)undo the stash file and start working on that again.

To view the stashed files

git stash list

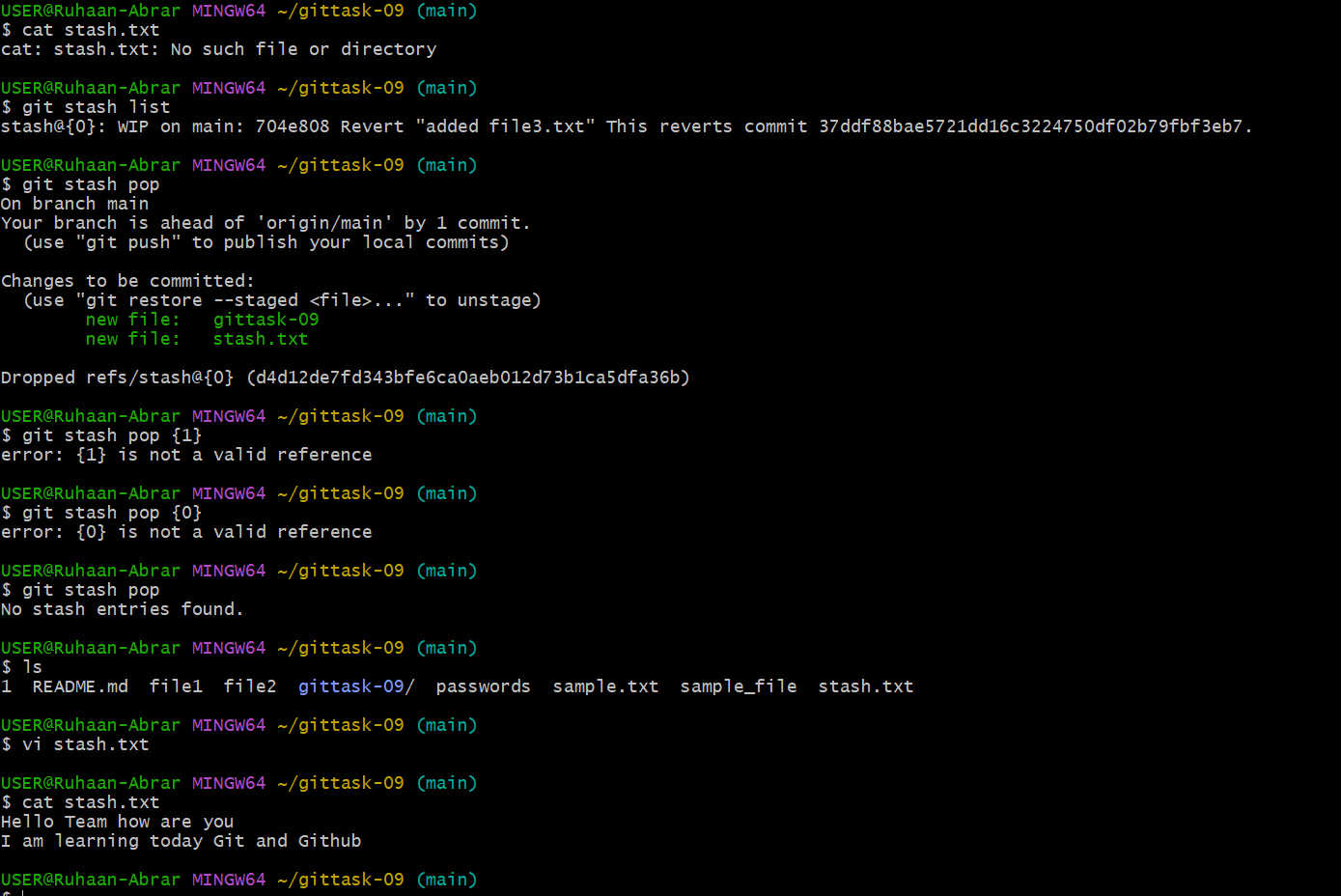
to remove or undo the stash

git stash pop

to continue working on that file

git add filename

git commit -m "started working on filename"



17)generate a ssh-keygen and configure into github.

Generate a SSH key in gitbash using

ssh-keygen

save the the public key in ~/.ssh/id\_rsa

ls to list the files in .ssh folder

cat id\_rsa.pub

copy that key without any spaces

Go GitHub and SSH keys section

Click on New SSH Key

select SSH

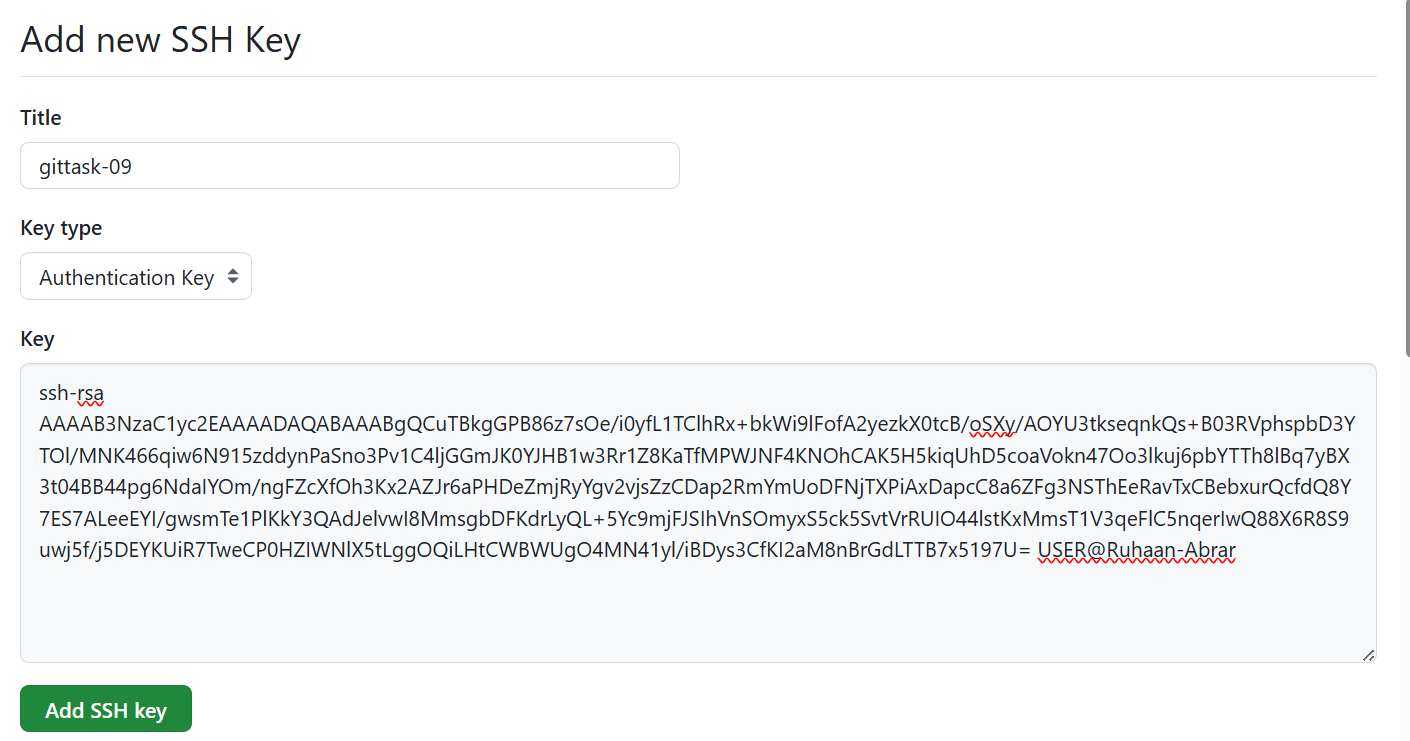
and paste the key there

and check if the connection is successfully configured or not

using git remote -v

This process is mandatory when we are pushing code to the central repo to avoid authentication





18)configure webhooks to github.

Go to GitHub and select your repository and select the SETTINGS

Create a webhook

select webhooks

Add webhook

Enter the payload URL, where GitHub should send every event

select the content type

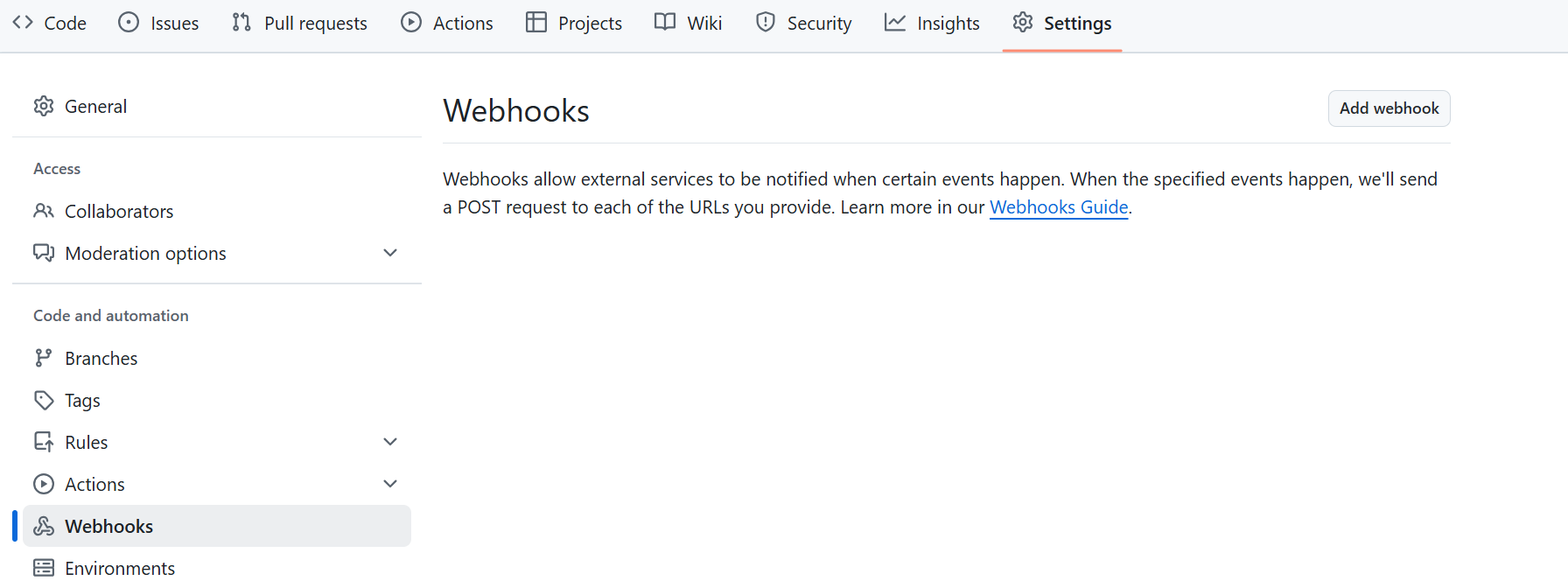
application/json

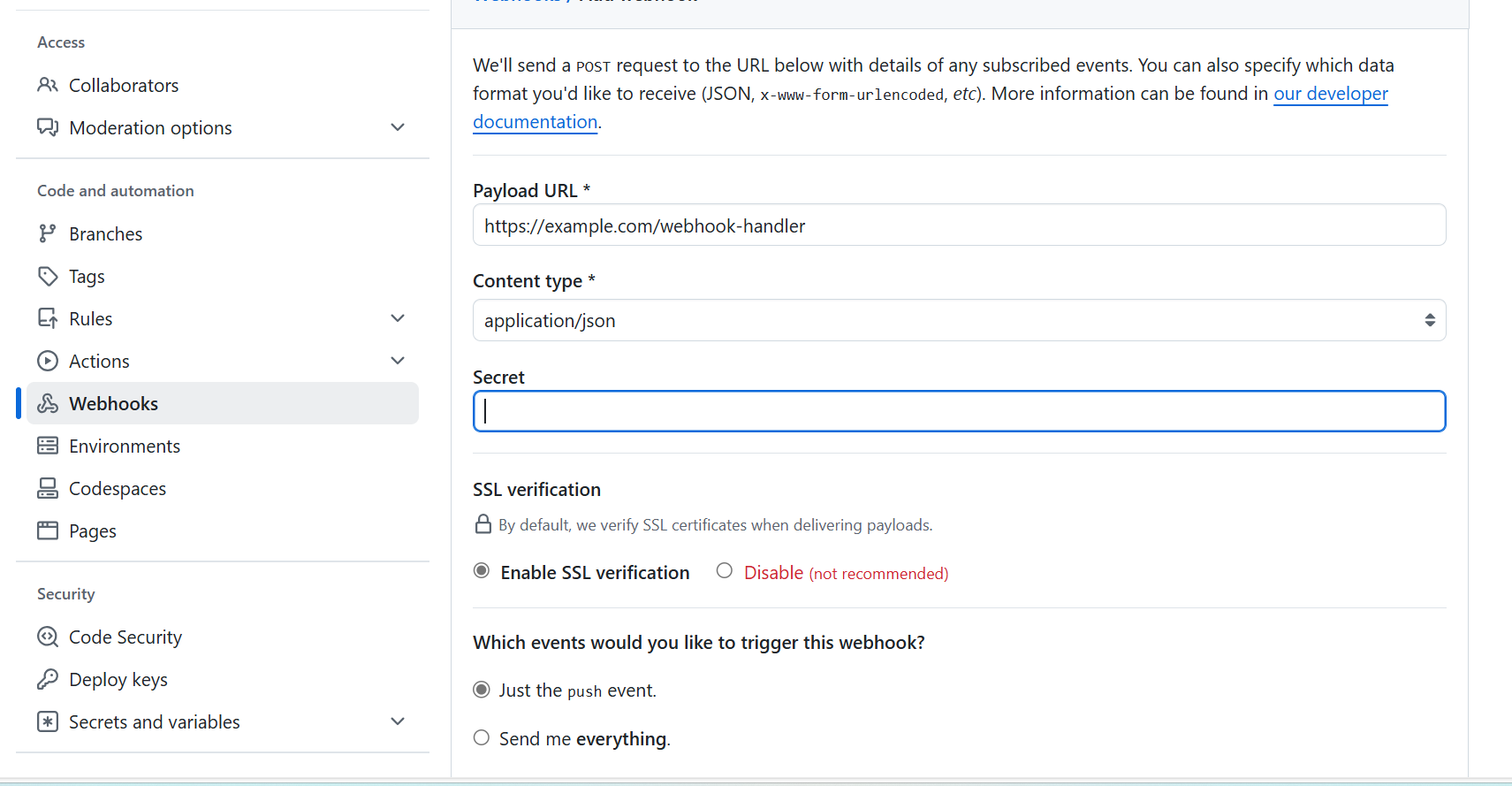
secret key is optional

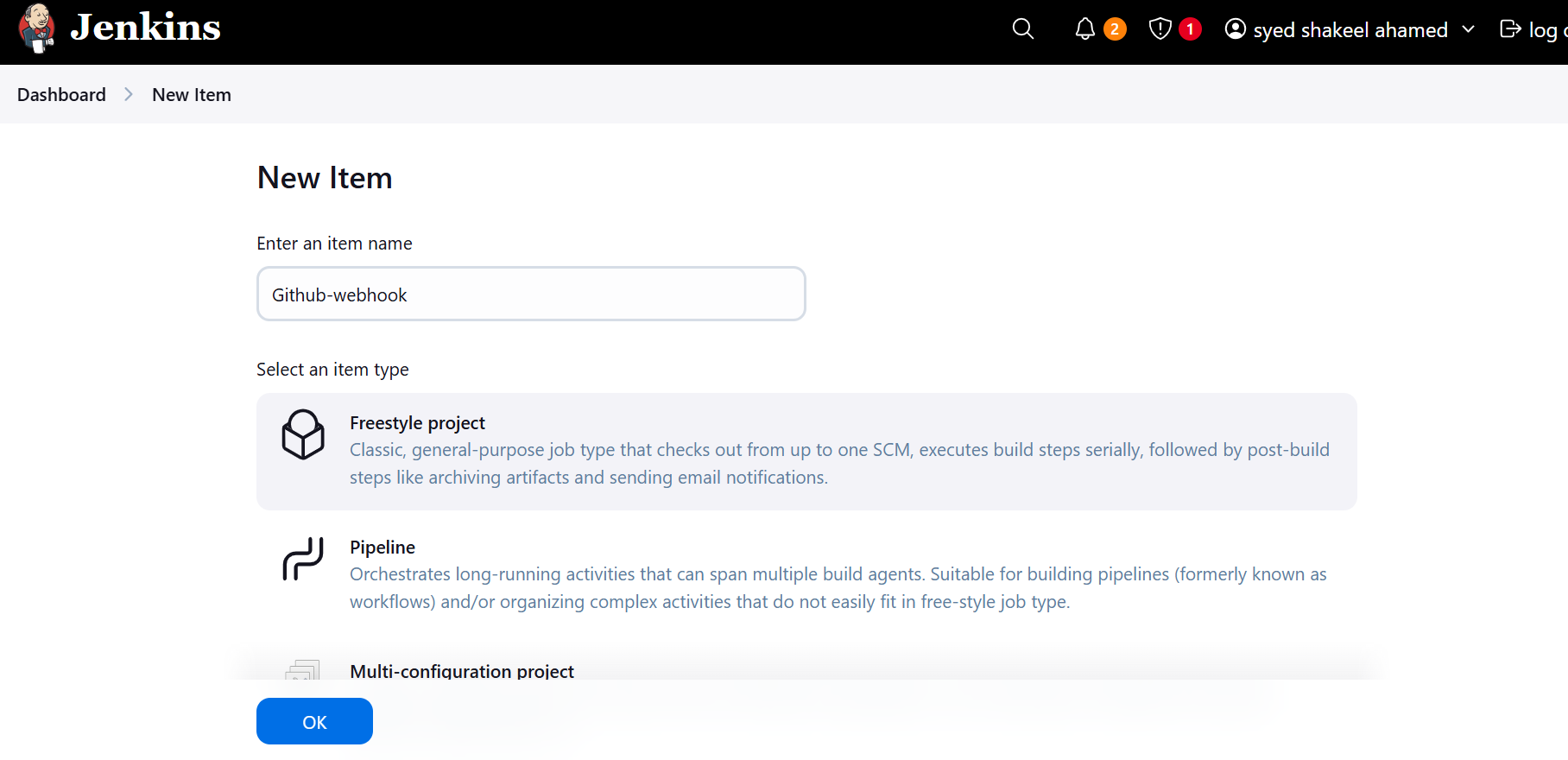
Add webhook

Test the webhook

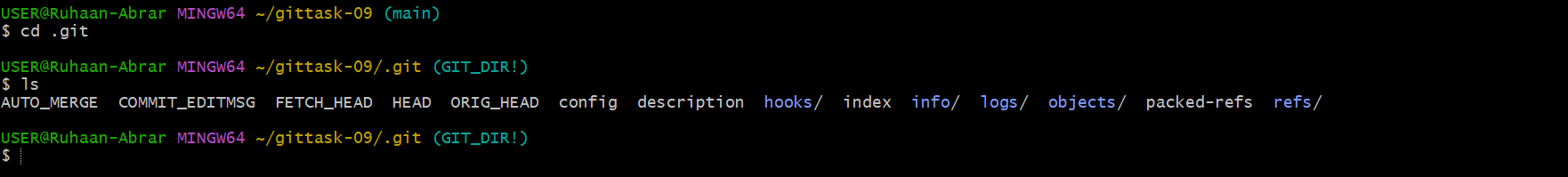
in that webhook page only check fore recent deliveries







19)basic understanding of .git file.



Git stores the metadata and object database for the project in this directory like: Remote information (to which remote server your project is connected) History of all local commits. Branch information (on which branch is your current project state (HEAD) pointing to)

Some important subfolders and files inside .git are:

* HEAD: Points to the current branch.
* config: Contains repository-specific configuration settings.
* refs/: Stores references to heads (branches) and tags.
* objects/: Stores all commits, blobs (file data), trees (directory structures), and annotated tags.
* logs/: Records actions performed in the repository (e.g., branch checkouts).

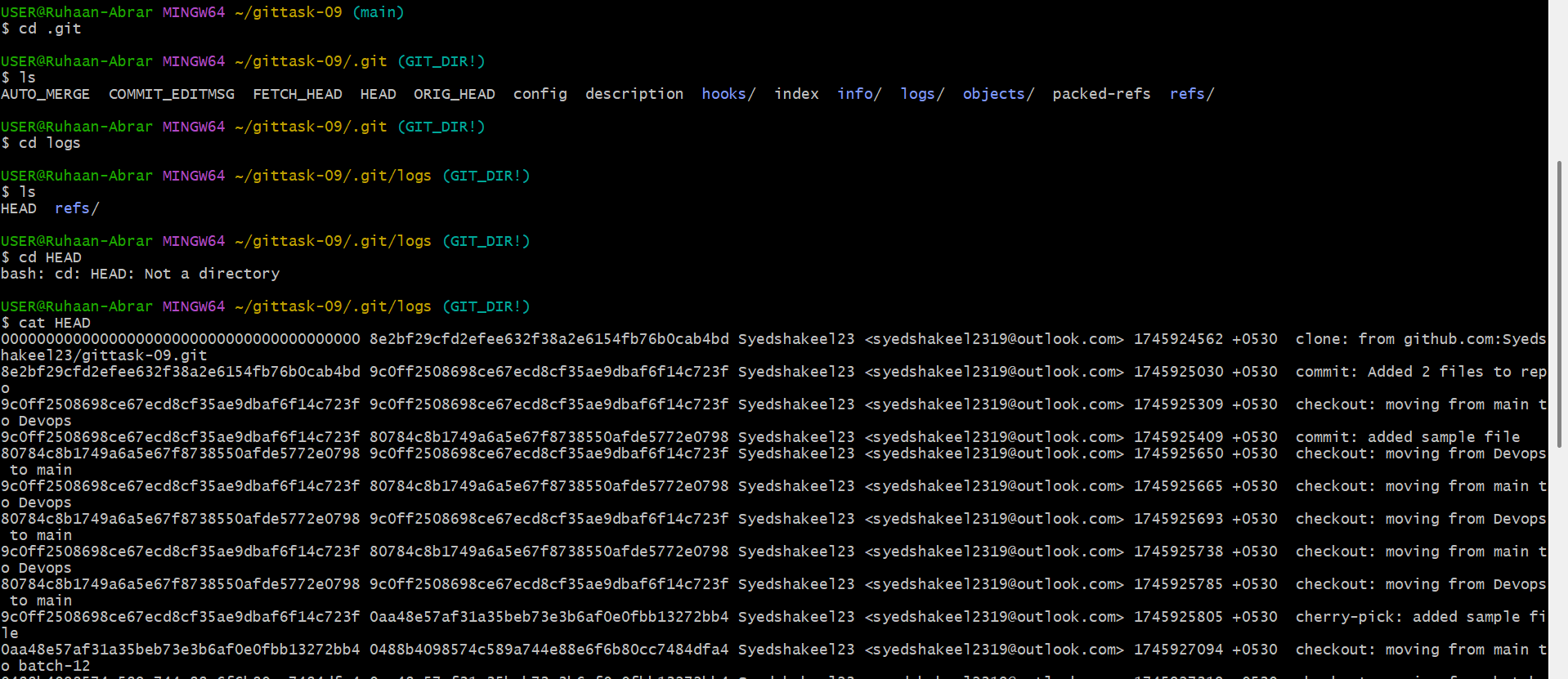
20)Check all the logs of git.

git log

git log -1 (for first line)

TO view logs for a brach

git log branch-name



21)Rename the commit message.

git commit --amend -m "New commit renaming"

git push

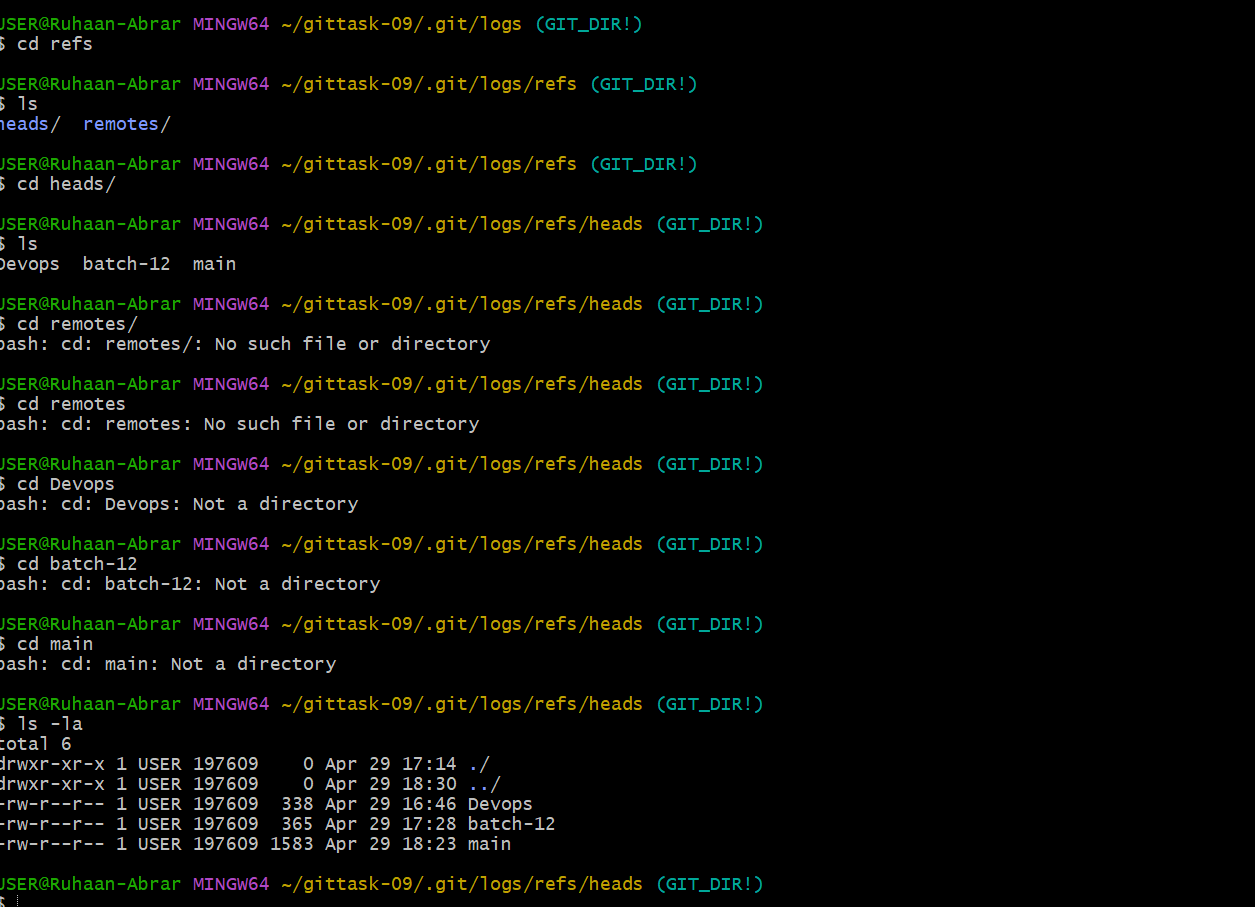
To rename older commits

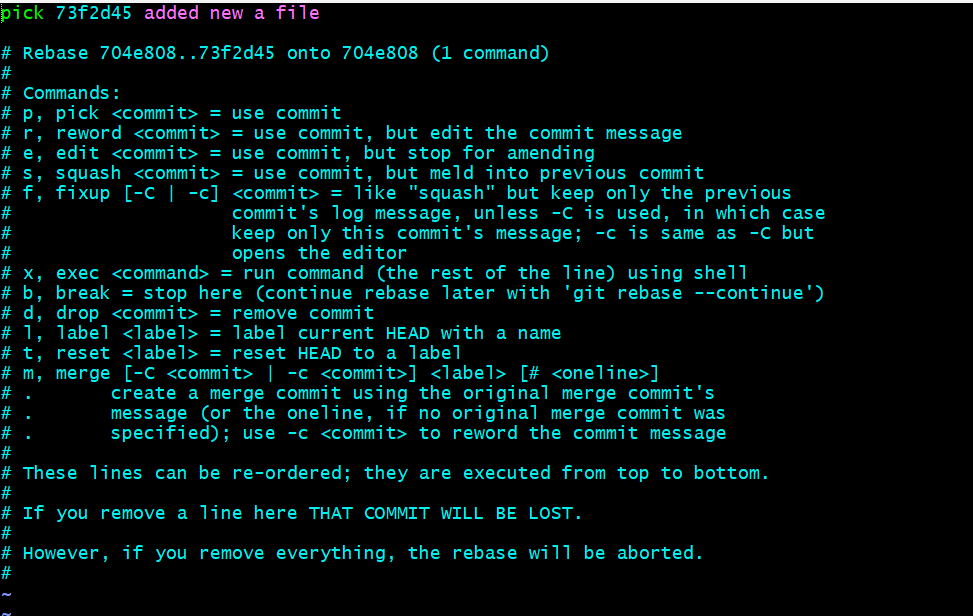
git rebase -i HEAD~4

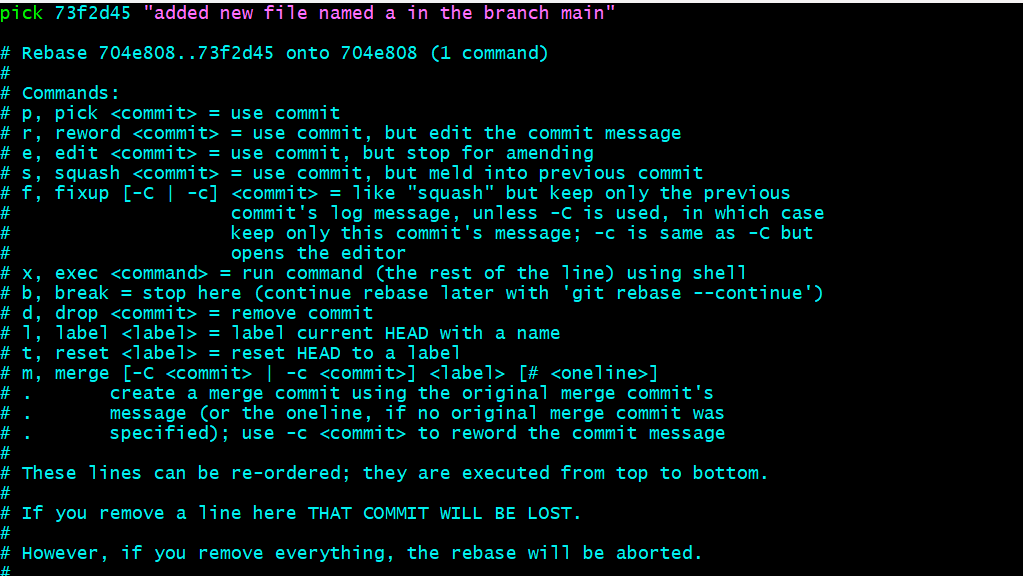
git commit --amend -m "4 older commits are renamed"

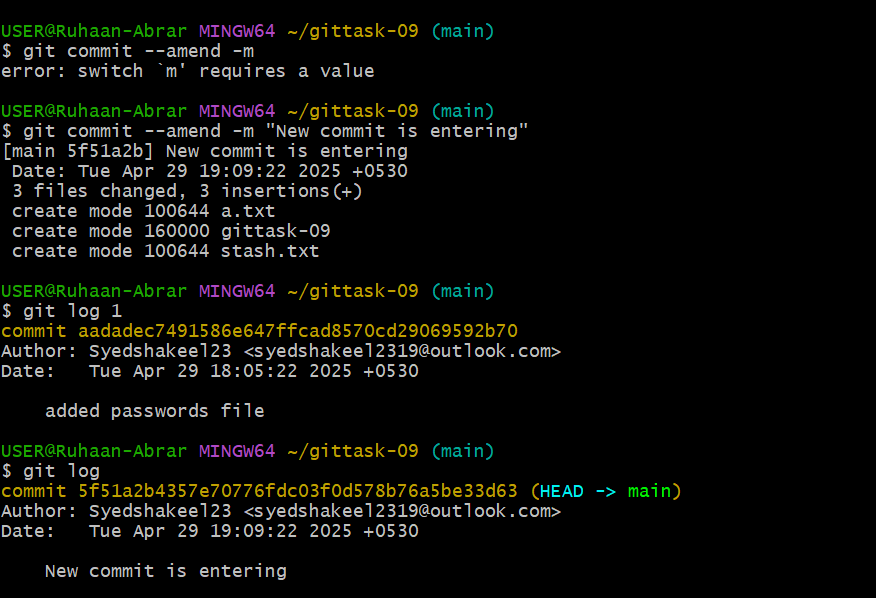
git rebase --continue

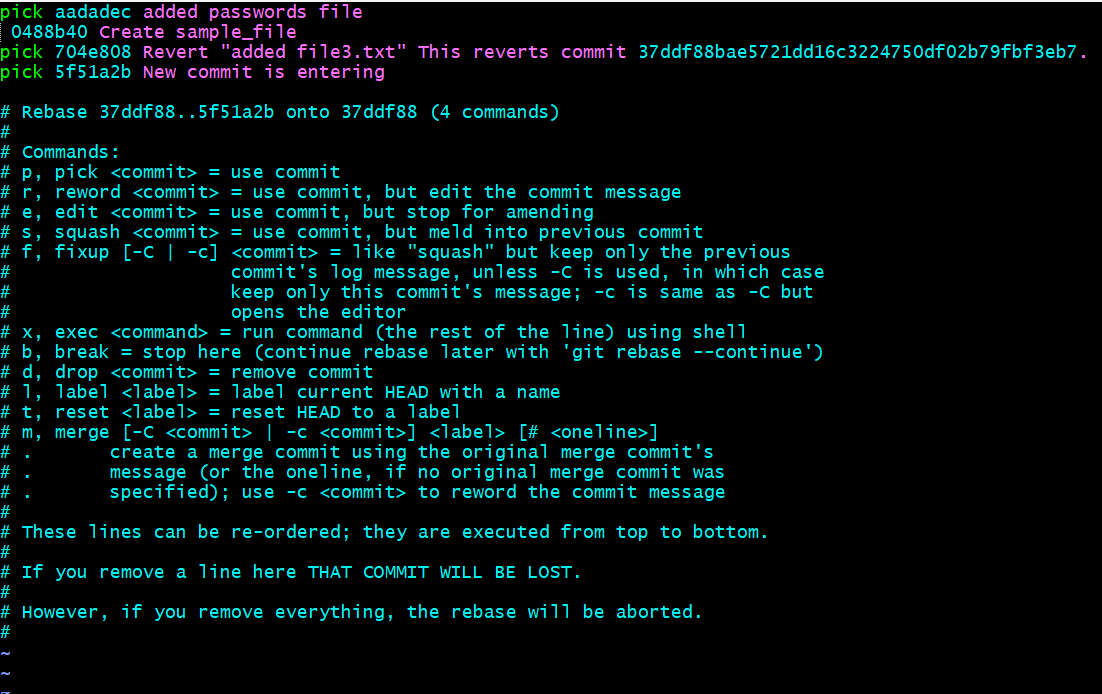
but this however not recommended to do any changes to commit messages











22)Merge multiple commits into single commit.

git rebase -i HEAD~4

so i have selected 4 commit messages

Modifying the commits using squash instead of pick

save the file

git push

Now you can see all 3 commit messages under one commit id

