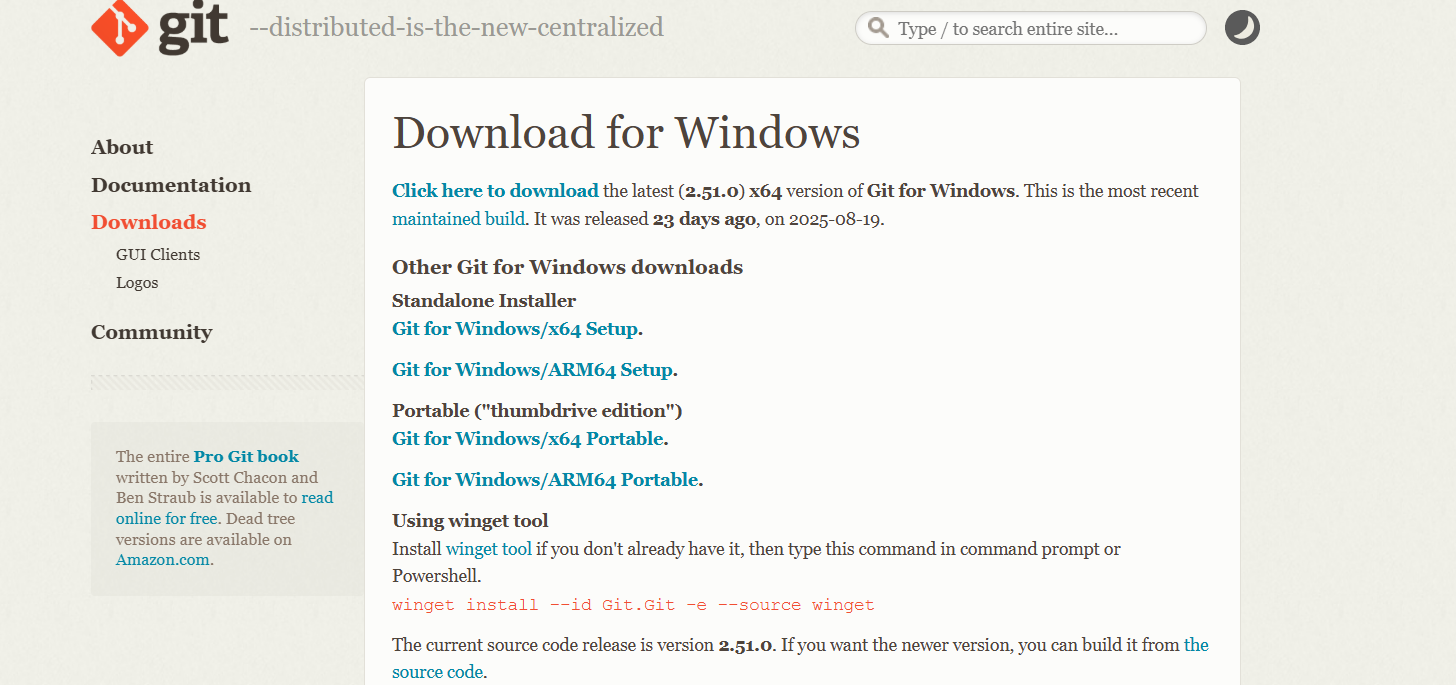
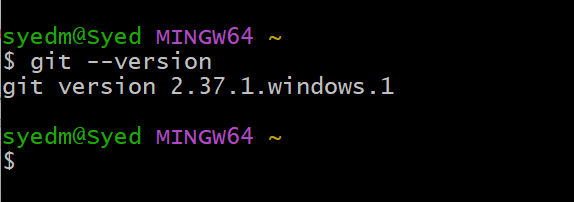
**GIT & GITHUB DAY-2:**

1. **Create a GitHub Account:**

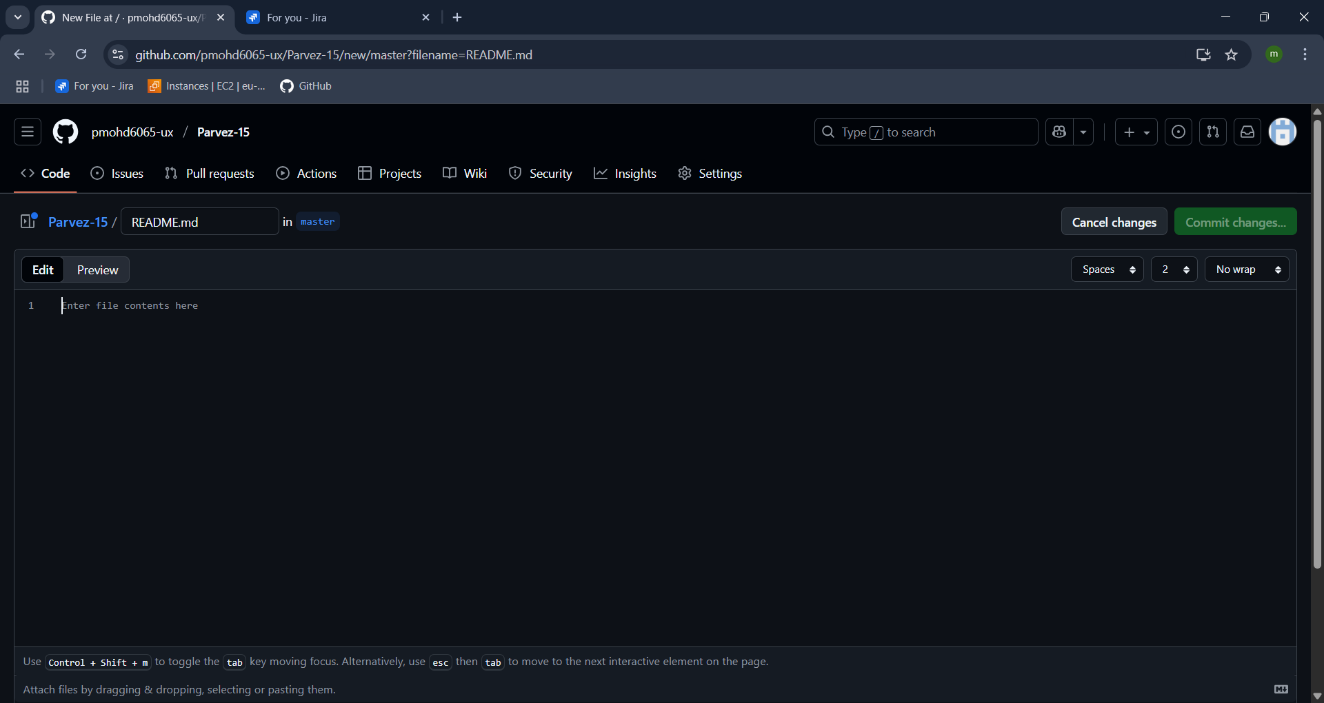
* Download Git from [Git - Downloading Package](https://git-scm.com/downloads/win)
* Run the installer with default settings



* Verify installation by opening Command Prompt or Git Bash and running:

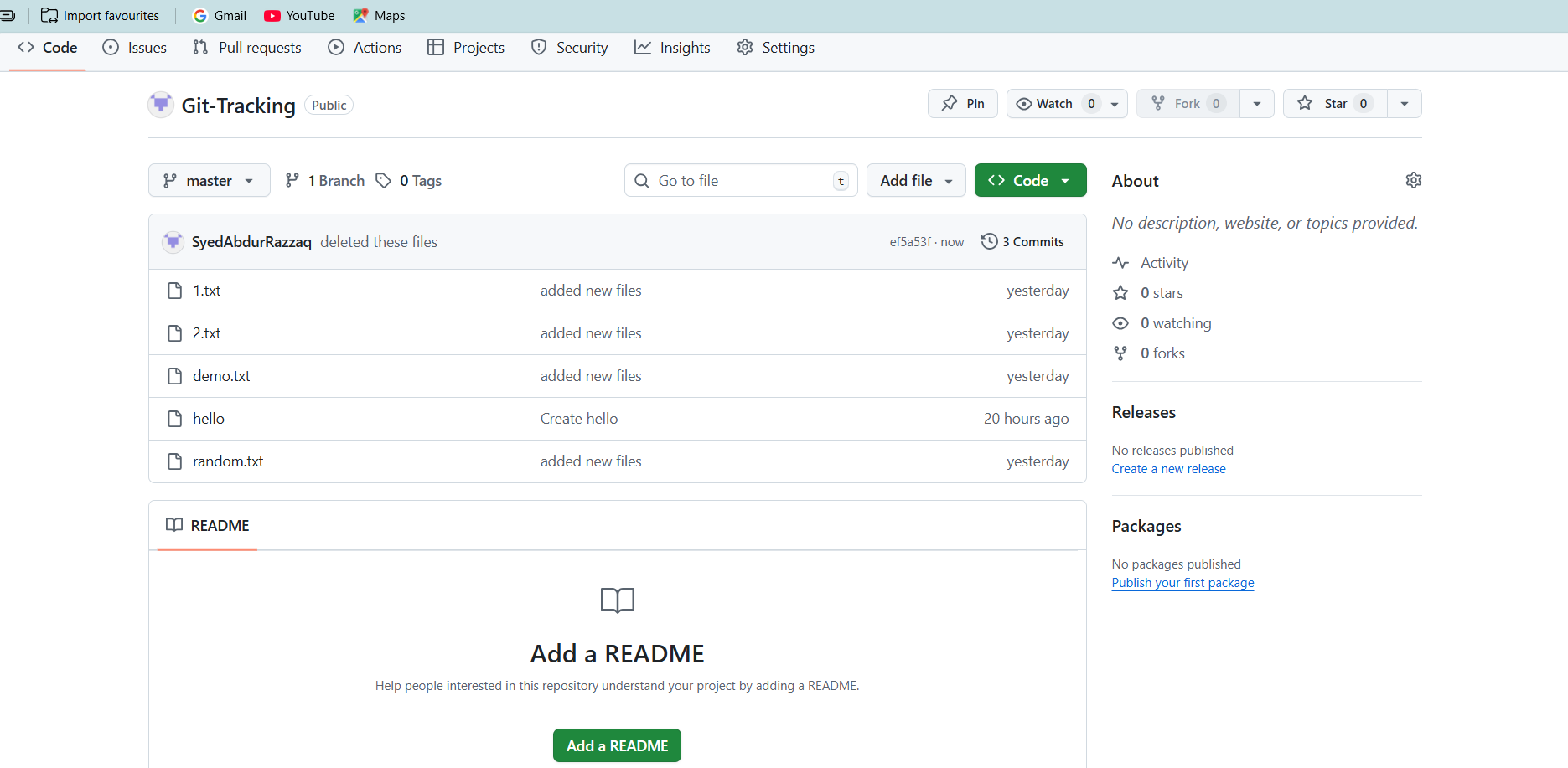


1. **CREATE A REPO IN GITHUB WITH README.MD AND .IGNORE:**

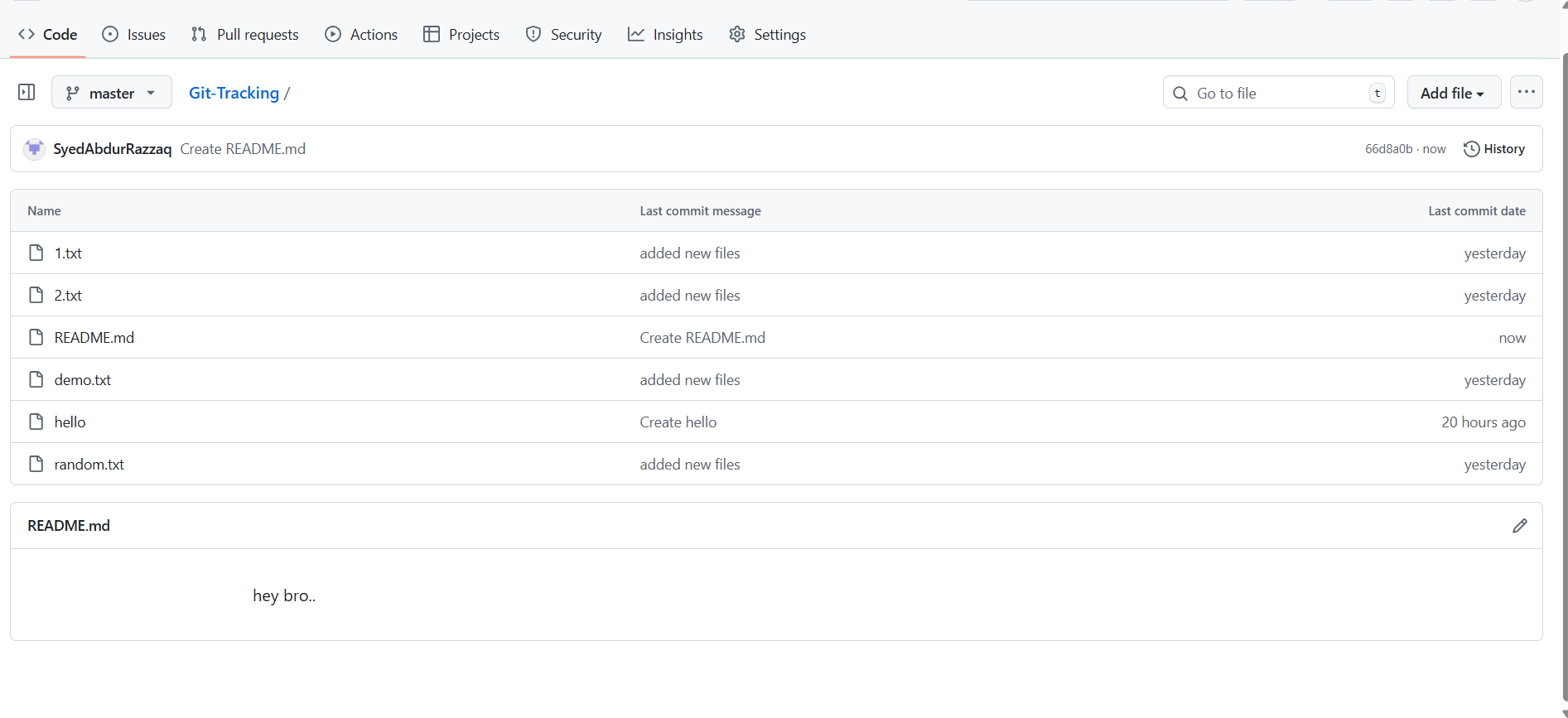
* To create README.md go to the repository in git hub scroll down you can see the add README click on it.



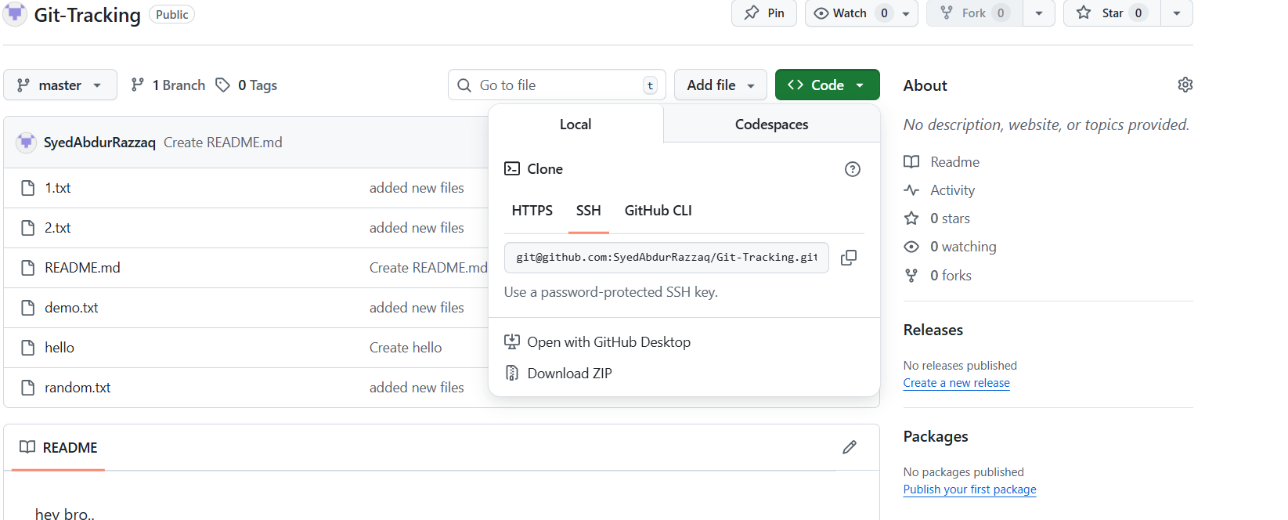
* Now the readme file gets open add the content for other user to understand the purpose of the project that you are working on.
* Then click on commit changes.



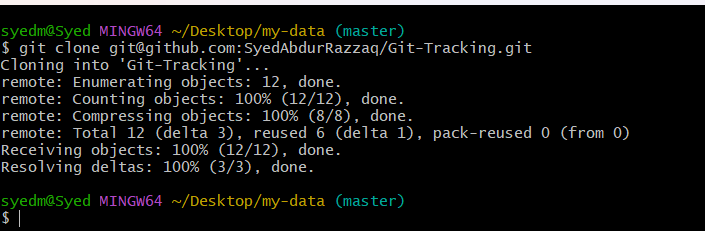
* You can see the readme.md file has created in the repository.



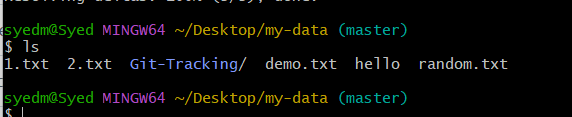
1. **Clone the created repo to local.**



* Go to the repo in github and click on the code copy the ssh url



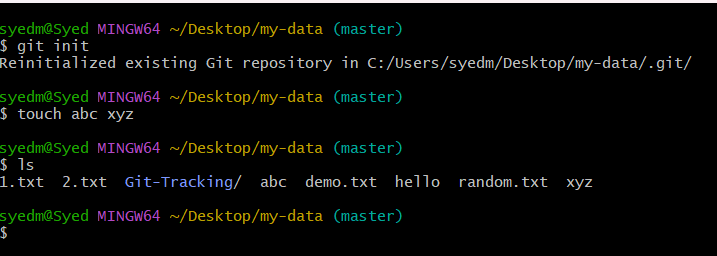
* Go to git in your local machine and go to the file which you are working
* Use: git clone -b “branch name” “url of the repo” – to clone the documents of the repo to the local machine.
* Note: git clone will download files with repository instead of latest changes made.



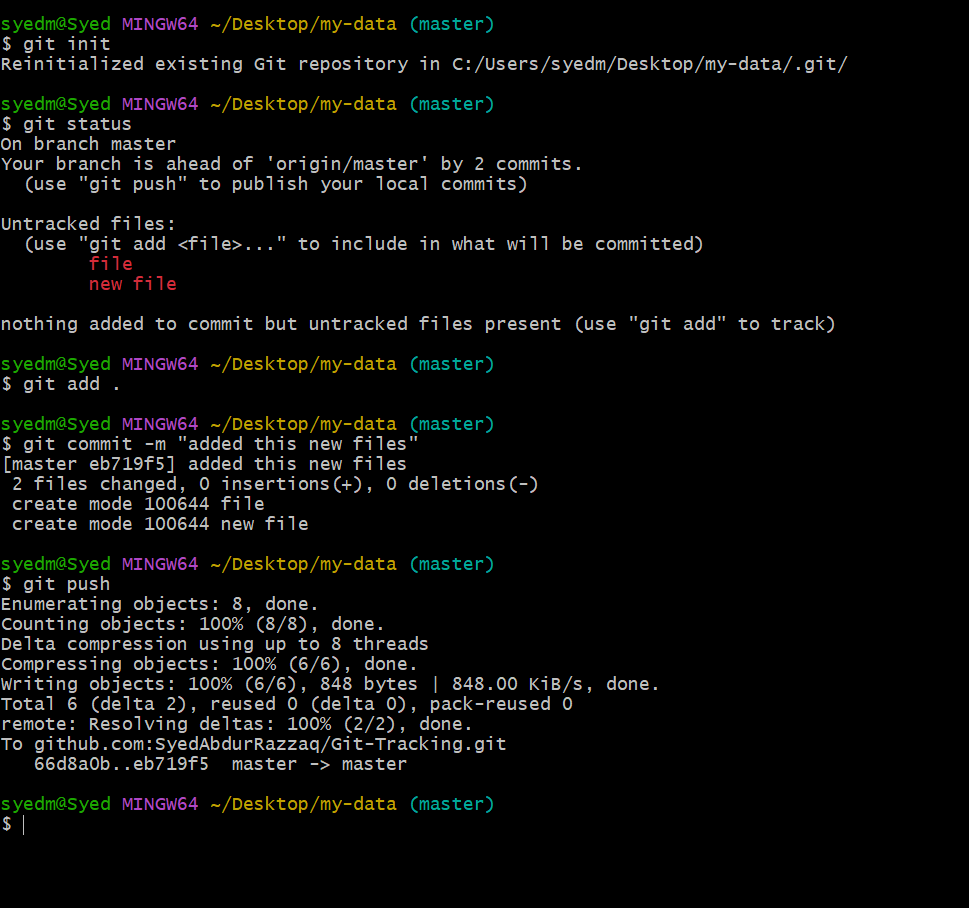
* You can see the name of directory name that is parvez-15 where all the files are downloaded.

1. **Create two files in local repo.**

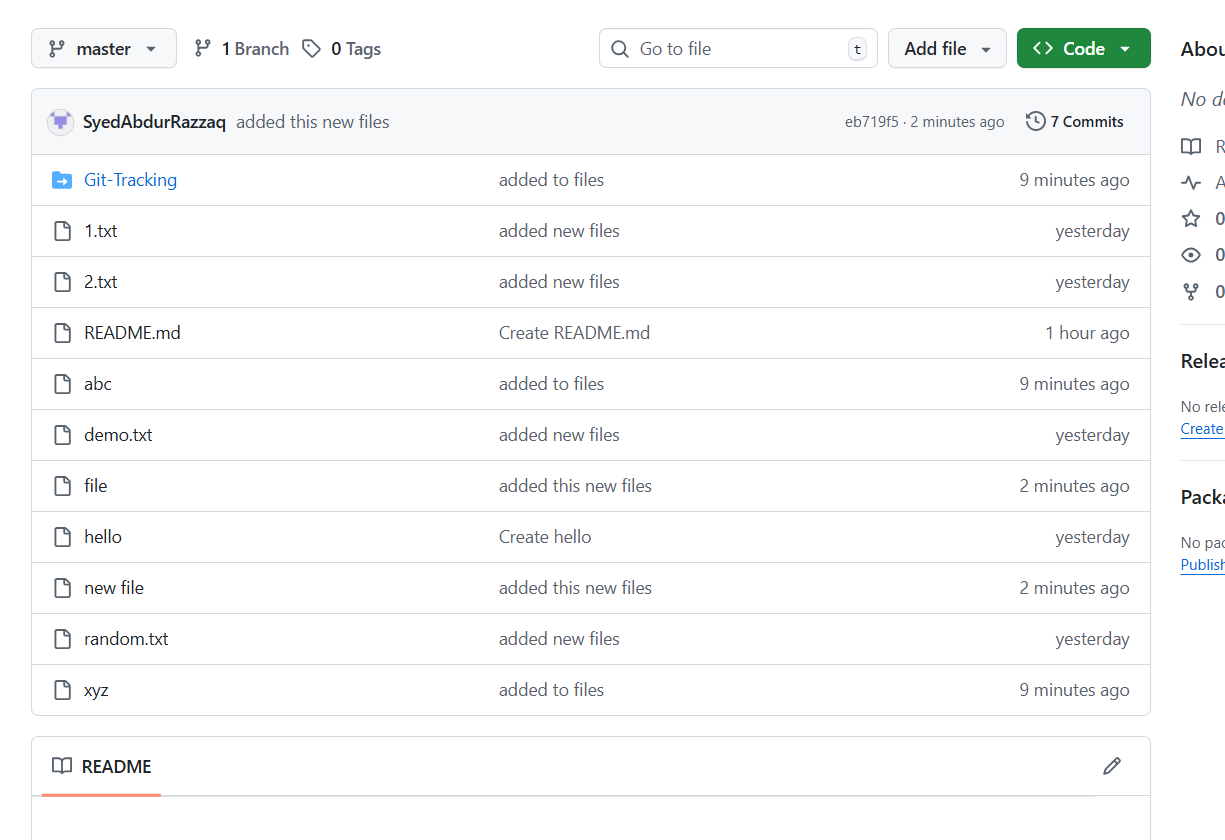
Create two files using touch command

You can see the files has been created in the local machine 

1. **COMMIT TWO FILES AND PUSH TO CENTRAL REPOSITORY.**

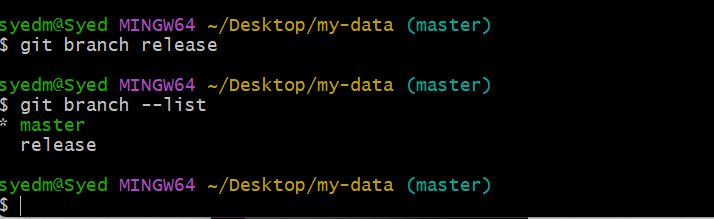


* Git add . : to track the repo
* Git status : to check the status
* Git commit -m “messege” to commit the changes you have made
* Check the status : git status – see working tree clean.
* Git log : to see the logs

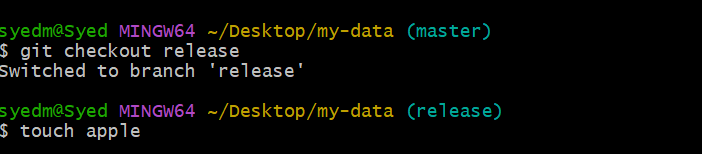


* Git push : to push the changes from local machine to the central repo
* You see in the central repo in git hub the file shas been added to the repo.

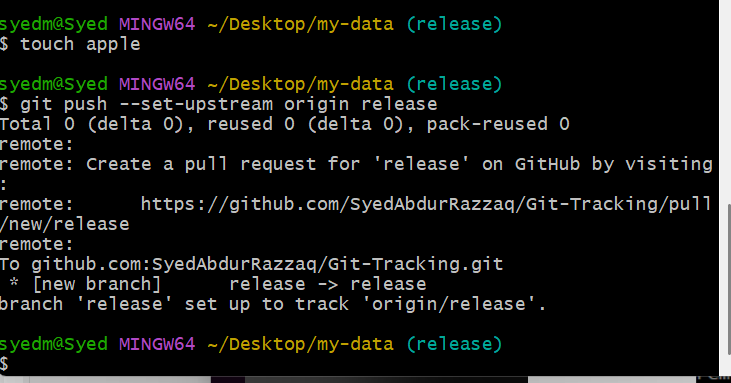
1. **CREATE A BRANCH IN LOCAL AND CREATE A SAMPLE FILE AND PUSH TO CENTRAL**



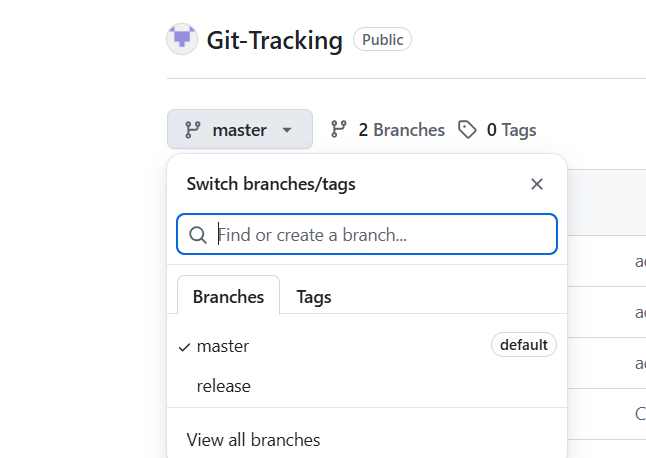
* Git branch “name” ; to create a new branch
* Git branch –list : to list the branches available \* means you are using the branch



* Git checkout “name” : to switch to the branch
* Touch: to create the file

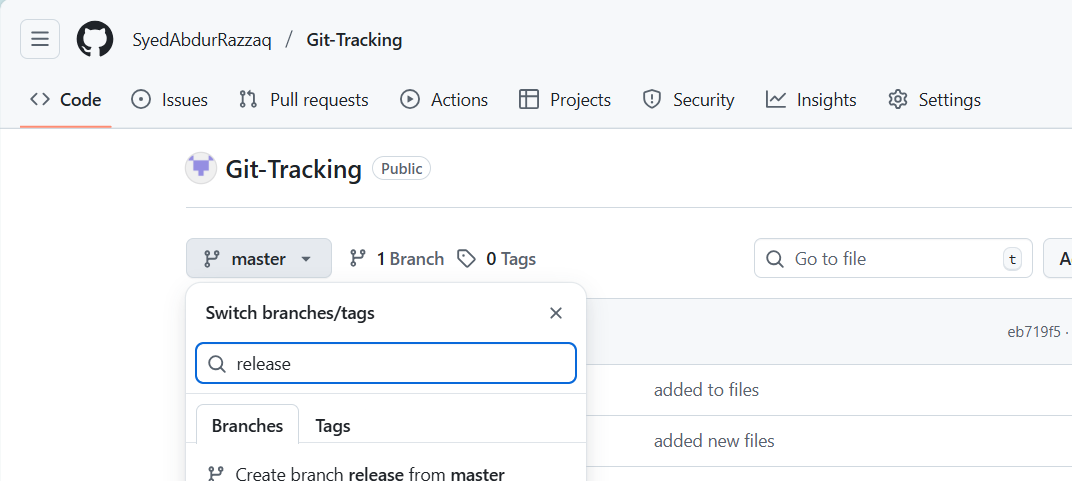


* To push the current branch and set the remote as upstream, use
* git push --set-upstream origin release

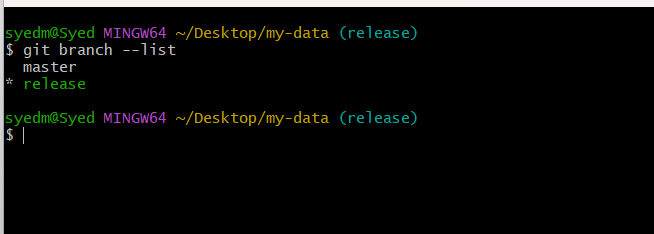


1. **Create a branch in github and clone that to local.**

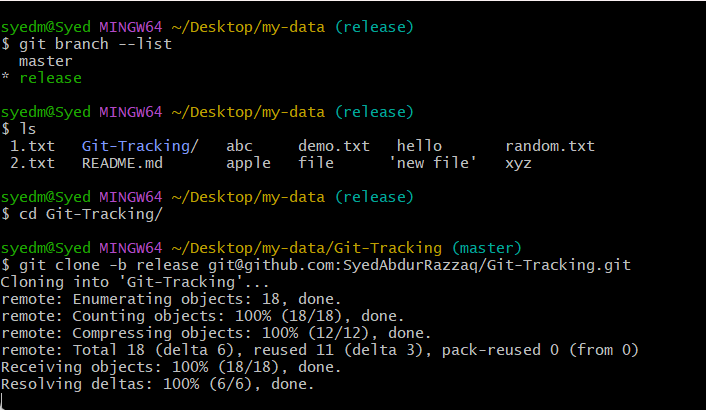
* Go to the git hub repo and click on branches

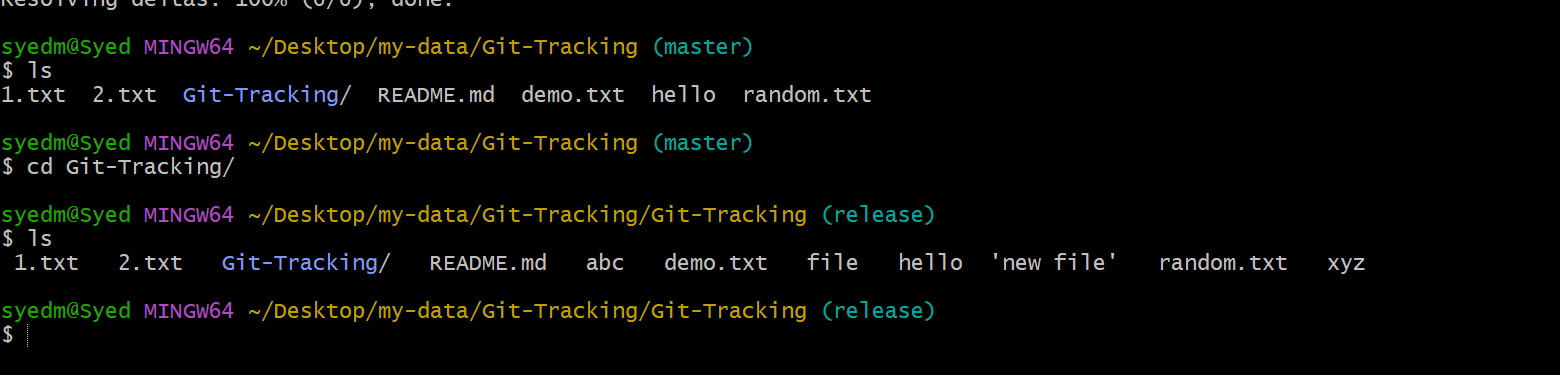


* Click on new branch and create new branch.

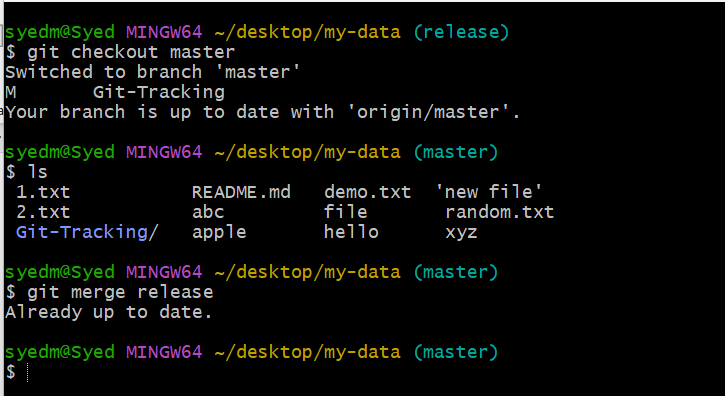


* Git clone -b “branch name” “ssh url” – to download the specific branch to local machine.
* You can see the bran name has changed to tests.



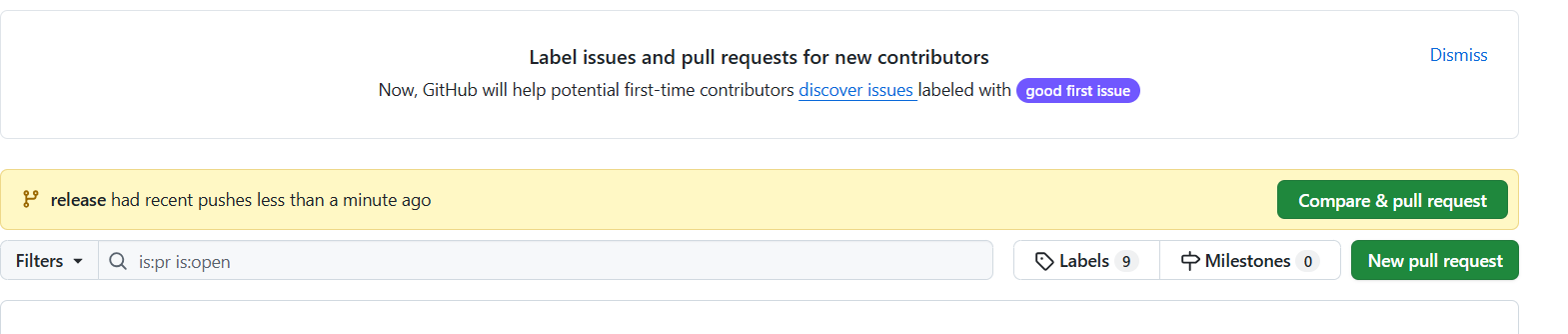


1. **Merge the created branch with master in git local.**



* Git merge “branchname” : to merge the branches with .
* You can see the similar files in tests and release branch in local machine.

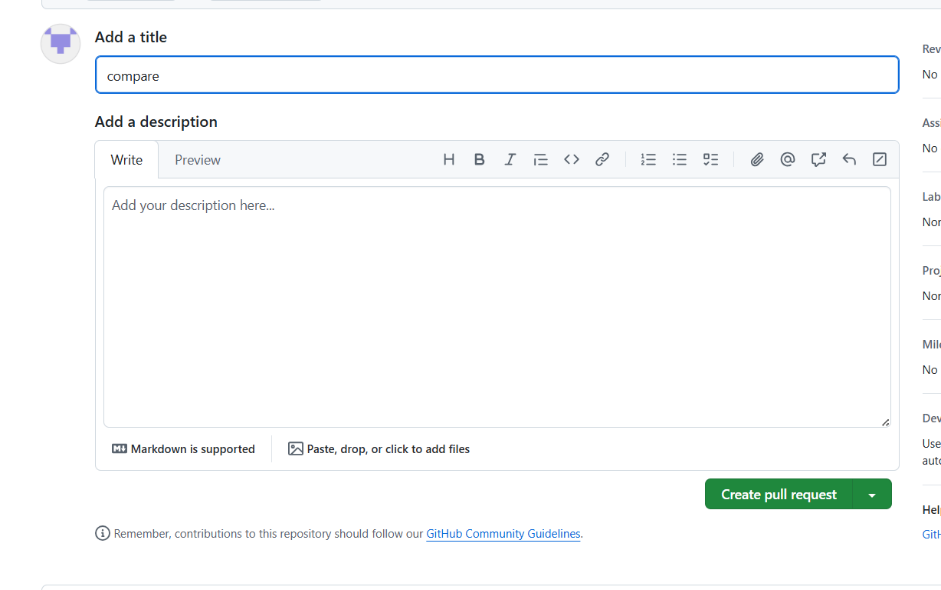
1. **Merge the created branch with master in github by sending a pull request.**



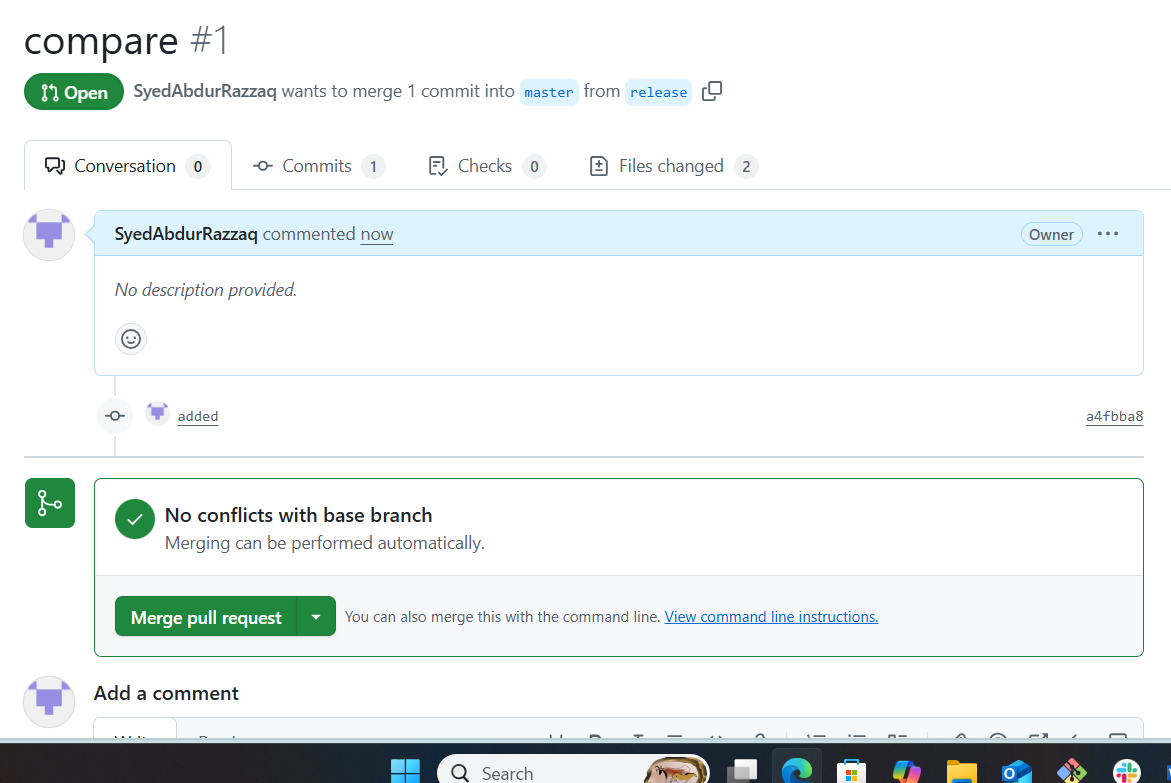
* Go to GitHub Repository

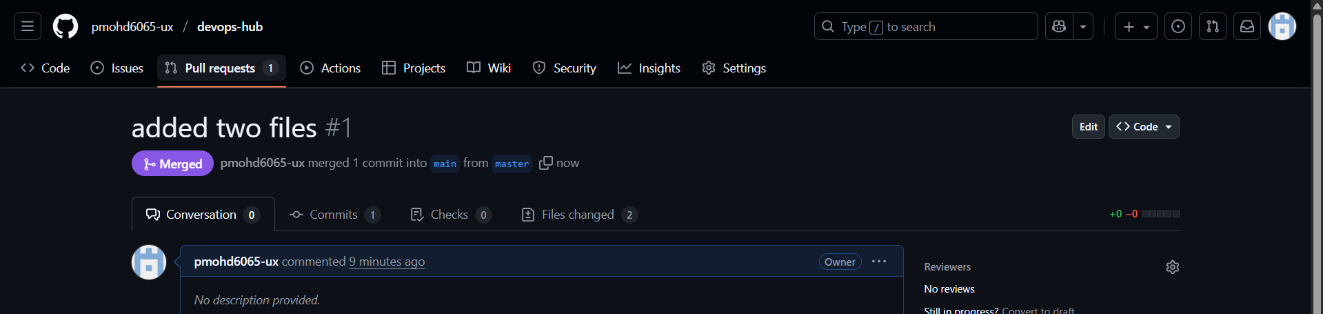
• Open your repo on GitHub.

• You’ll usually see a prompt like: “Compare & pull request” for your branch



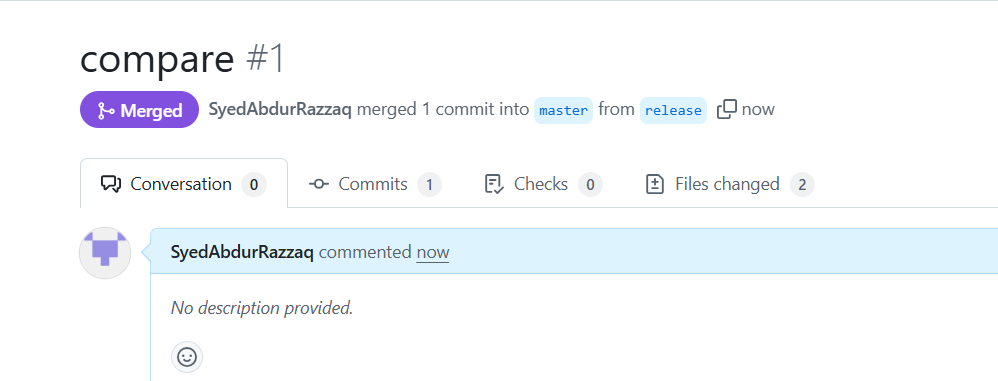
* Click “Create pull request”



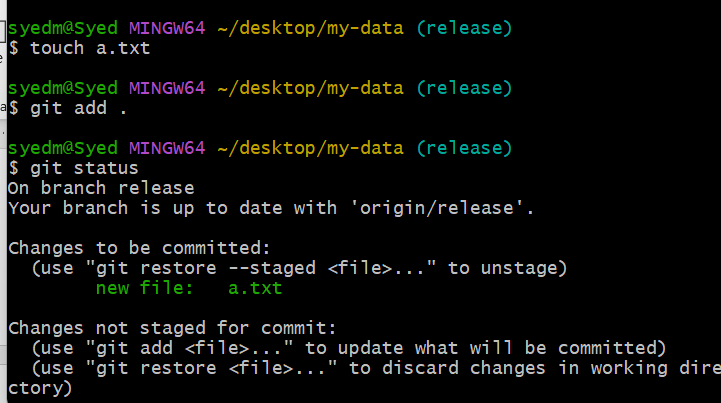
* Click “Merge pull request”.
* Confirm by clicking “Confirm merge”



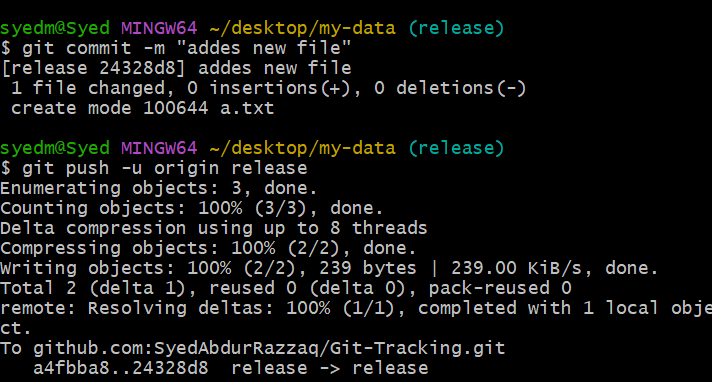
* You can see the files has been merged.



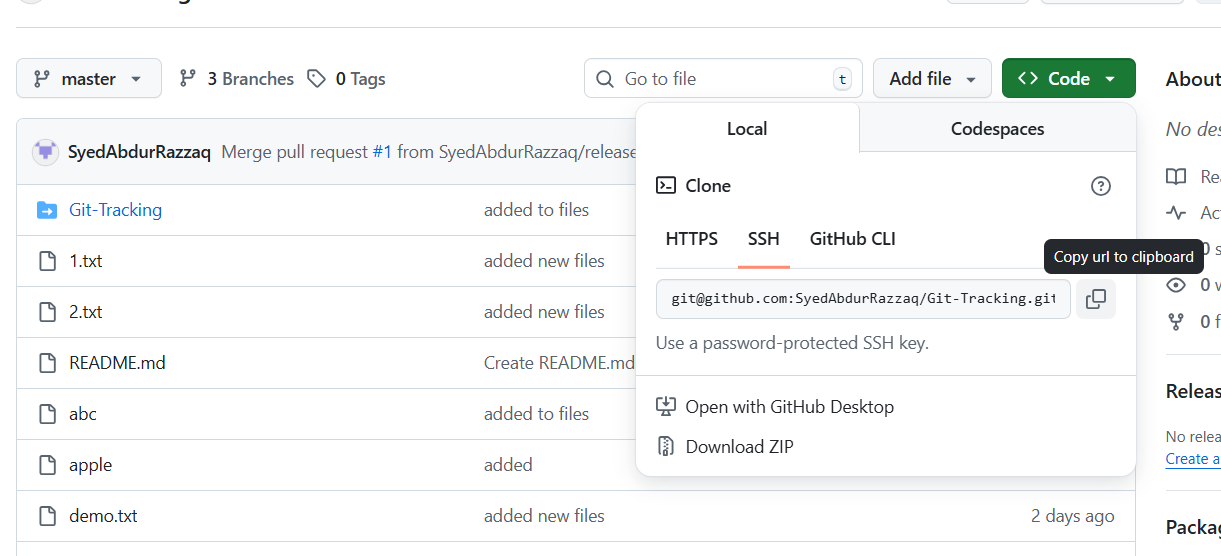
1. **Create a file in local and send that to branch in github**



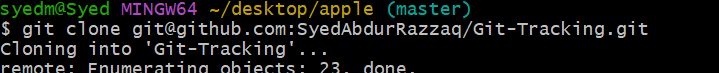
* create a file : touch a.txt
* Addthe file to tracker : git add .
* Check the status : git status
* Git commit -m “messege “ commit the changes and add messege.
* Git push -u origin master : to push the file from local to the branch in git hub.



1. **Clone only a branch from github to local.**



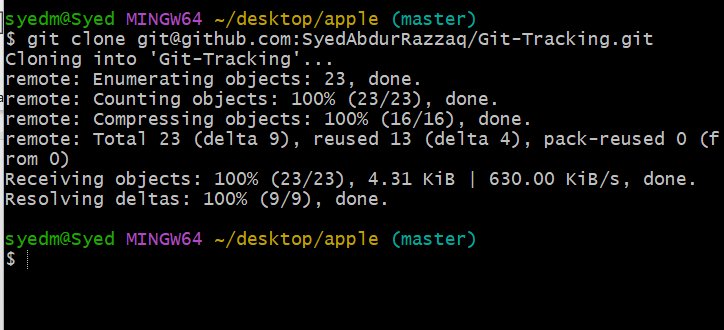
Copy the ssh url of the branch you want to clone



* Git clone “ssh url” : enter in git to clone the branch
* Then it clones and creates a directory with the repo name

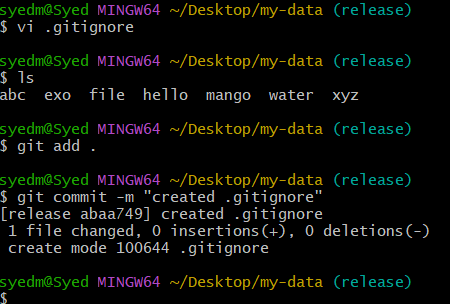


* You can see the branch has been cloned.

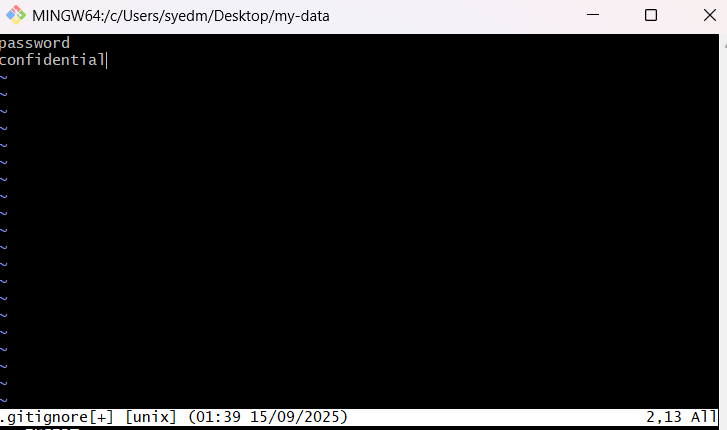


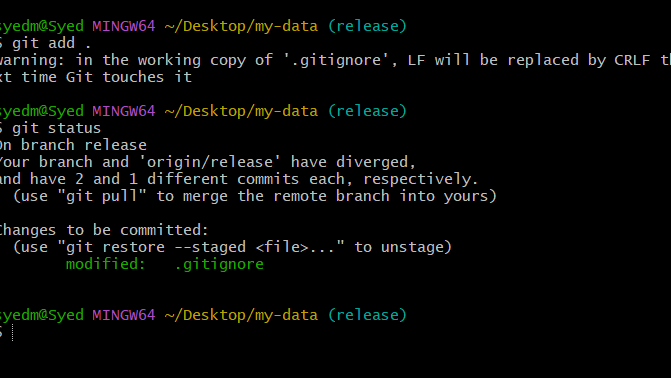
1. **Create a file with all passwords and make that untrackable with git.**

* Create ignore file : vi .gitignore
* Git add . : to update the git
* Git commit -m “messege” : commit the changes made.



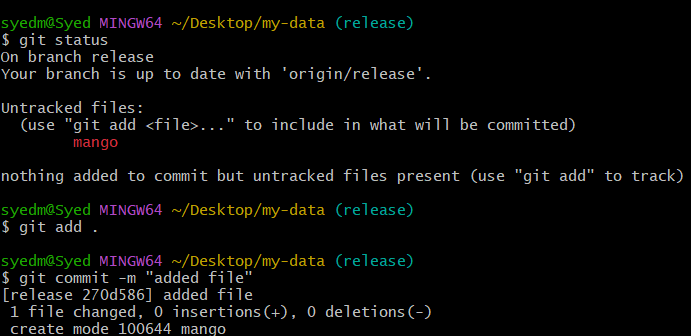
* Vi secretfile : create a secret file.
* Vi .gitignore : add the file name that you want to get untracked
* Save and exit the file



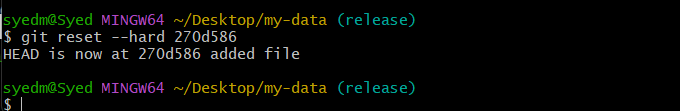


* git add . : to track the changes
* Git status : you can see the password file is being untracked.

1. Make a commit and make that commit reset without savings changes.

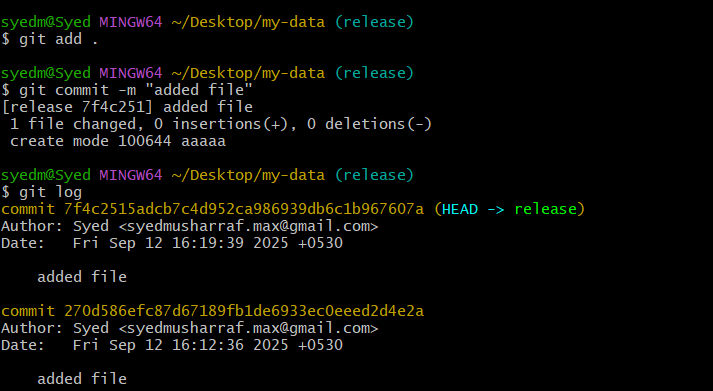


* Git add . : to make the changes trackable
* Git commit : to commit the changes ,Copy the git commit id

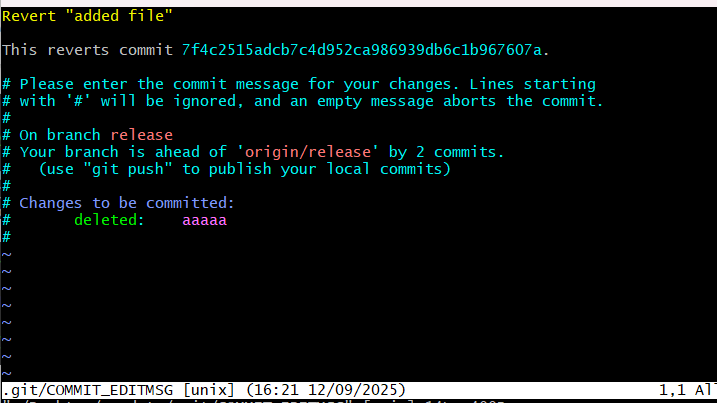


* Git reset - -hard commit id : to reset thecommit without saving changes

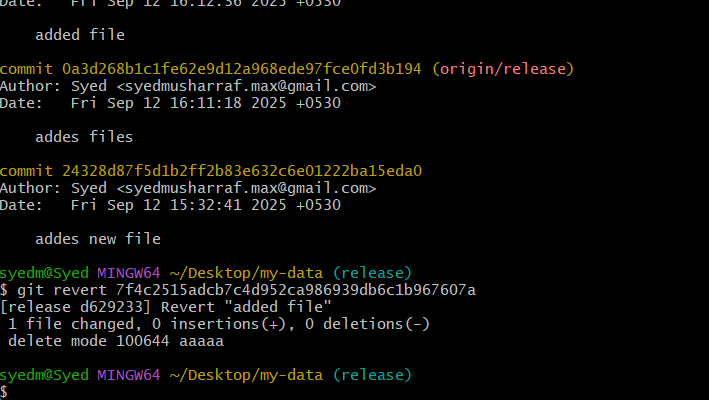
1. Revert a commited commit to the older version



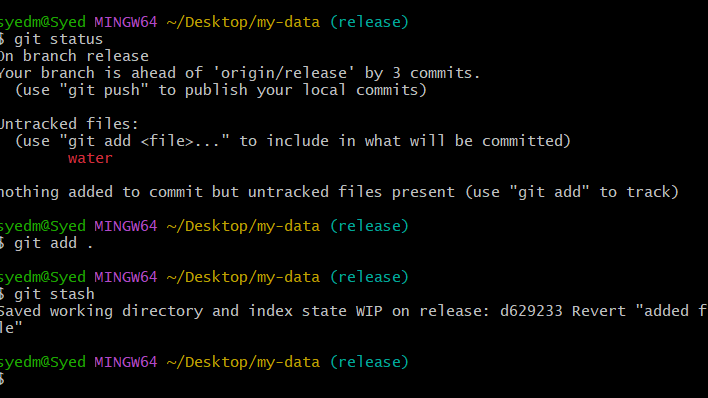
* Crete a commit : git commit -m “messege”



* Git revert <commitid> :revert the changes
* You can see the changes have been revert.



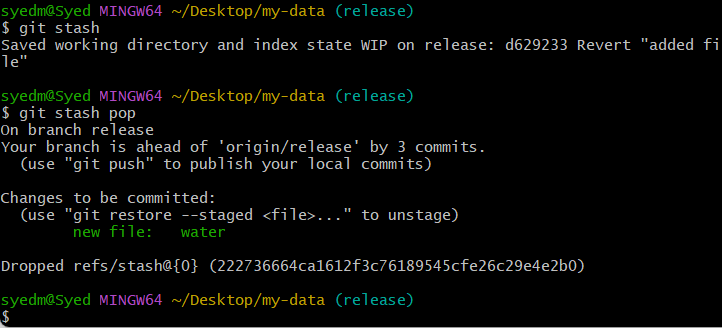
1. Push a file to stash without savings the changes and work on another file.



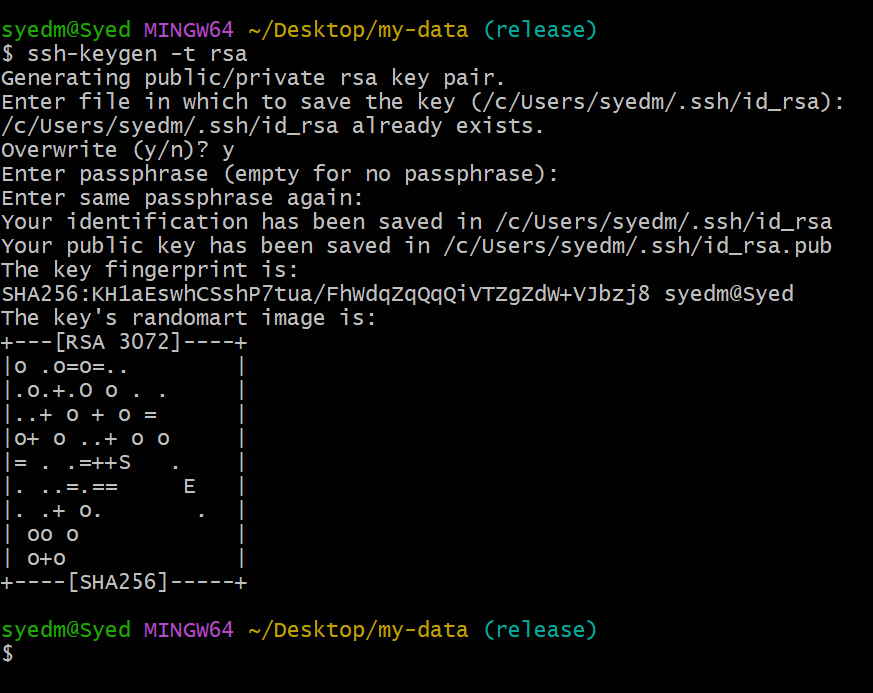
* touch new :create ne file work and save it
* git add .
* Git stash : move the file to the stash memory temperory

1. Undo the stash file and start working on that again.

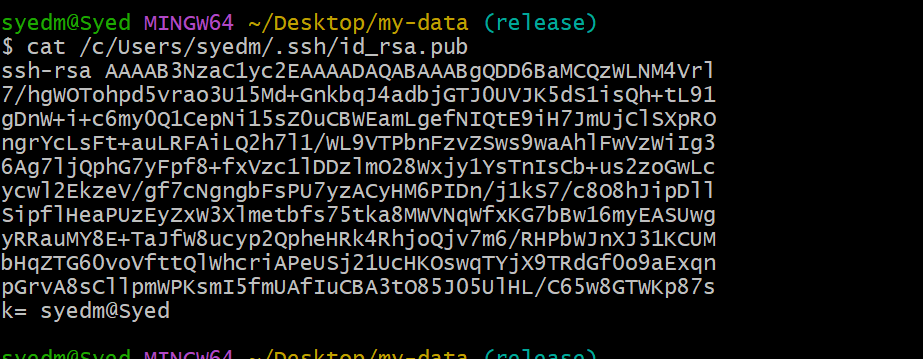
Git stash pop :to undo the stash file and work on that



1. Generate a ssh-keygen and configure into github.



* Ssh-keygen -t -rsa : to generate sshkey
* The key will be created local machine



* Cat “path/to/key” : to view the sshkey
* Copy the key

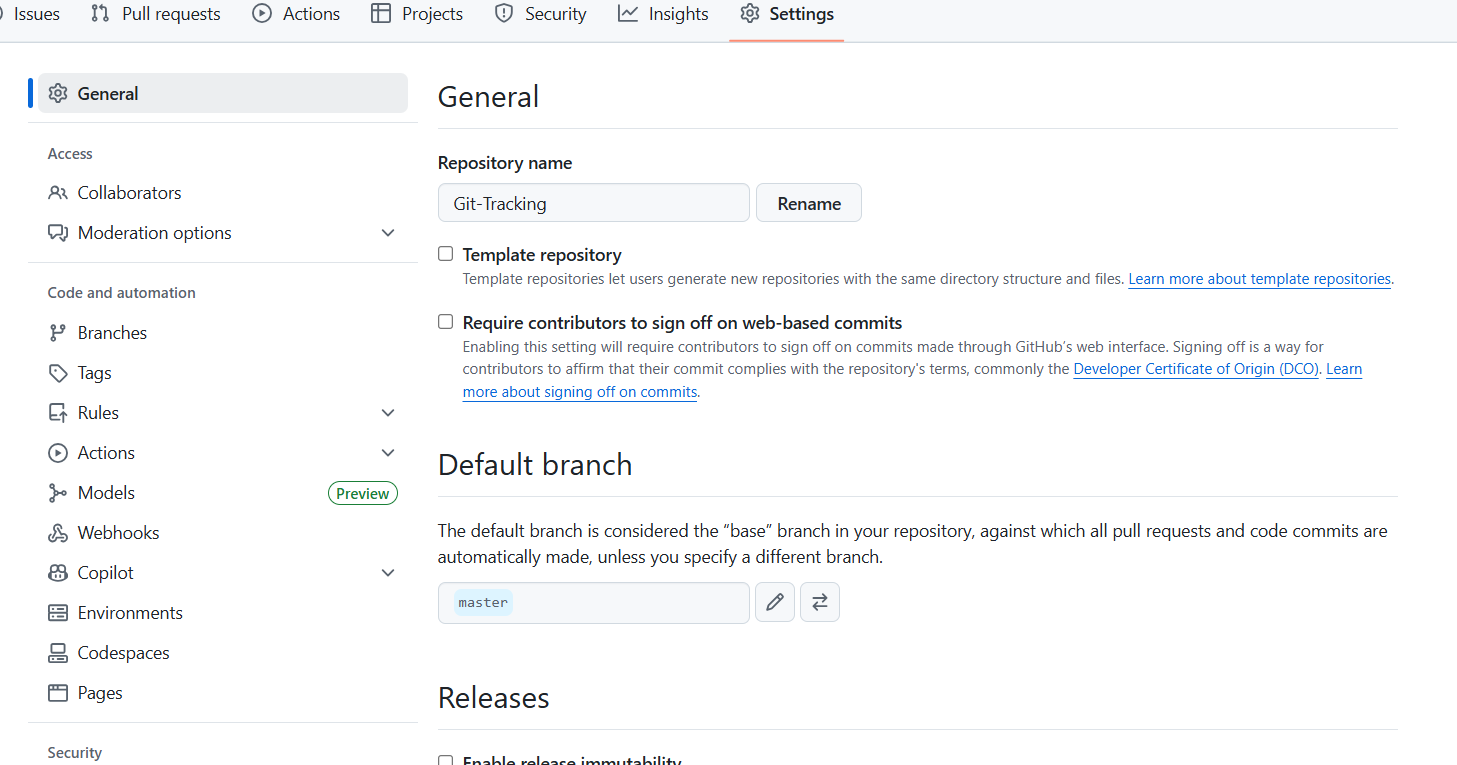


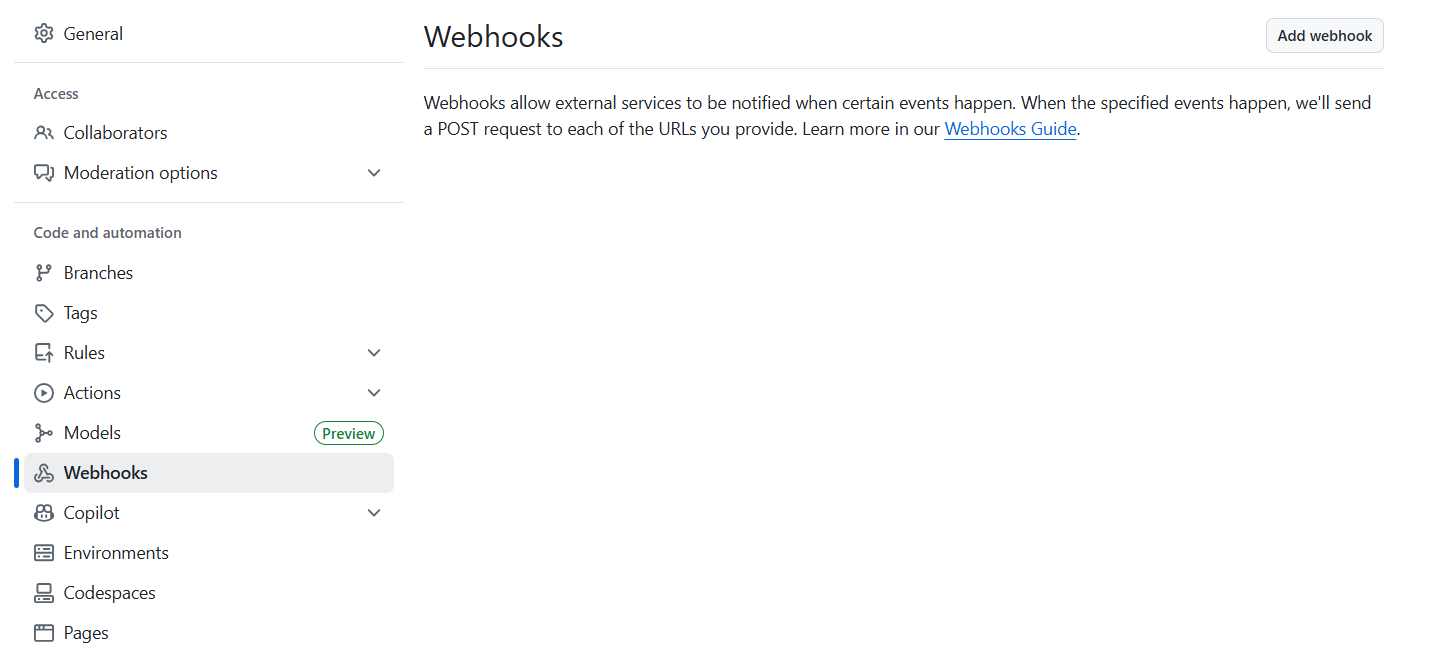
* Add ssh key and confirm through the code and configuration is done.

1. Configure WEBHOOKS TO GITHUB

You can create webhooks to subscribe to specific events on GitHub that occur in a repository, organization, GitHub Marketplace account, GitHub Sponsors account, or GitHub App.

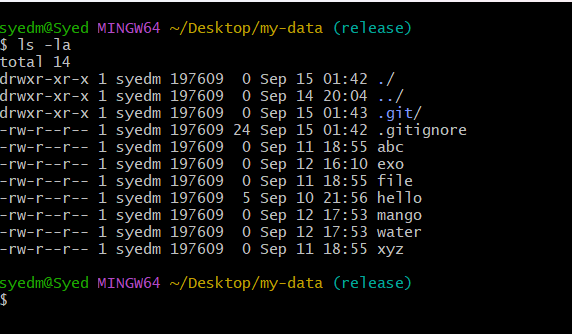
You can create a webhook to subscribe to events that occur in a specific repository. You must be a repository owner or have admin access in the repository to create webhooks in that repository





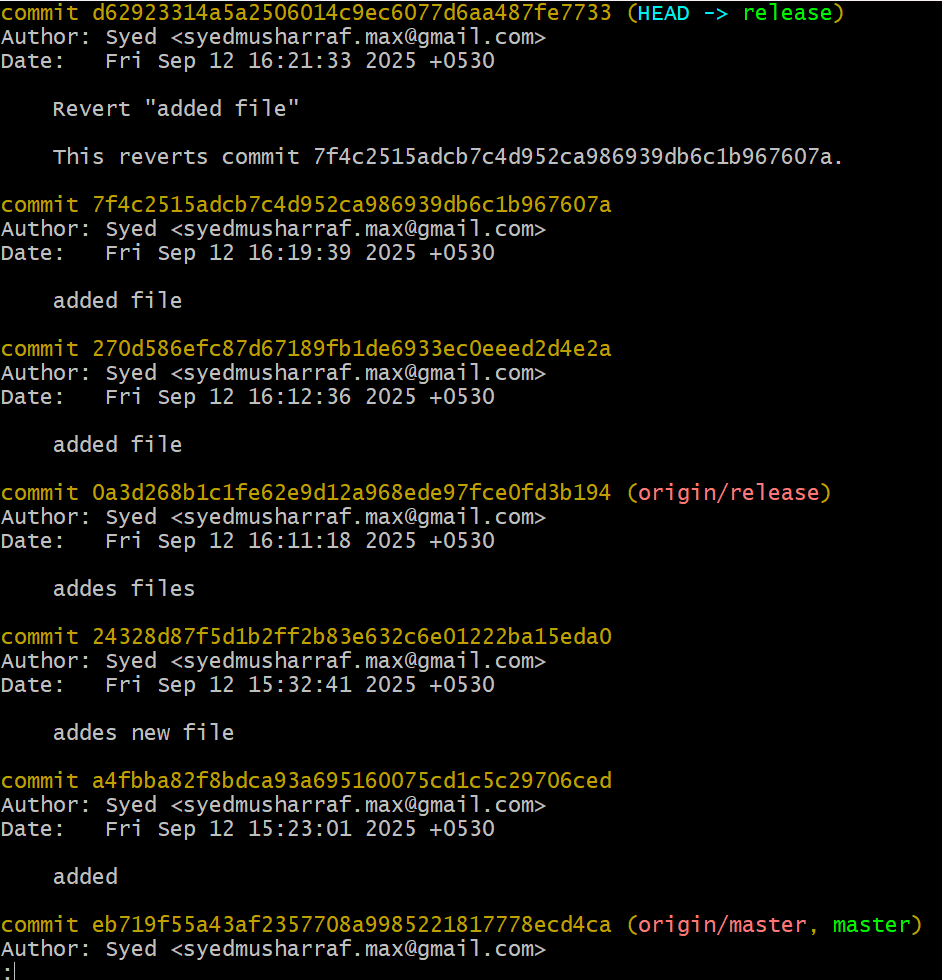
* Under your repository name, click  Settings. If you cannot see the "Settings" tab, select the  dropdown menu, then click Settings.
* In the left sidebar, click  Webhooks
* Click Add webhook.
* Under "Payload URL", type the URL where you'd like to receive payloads.
* Optionally, select the Content type drop-down menu, and click a data format to receive the webhook payload in.
* application/json will deliver the JSON payload directly as the body of the POST request.
* application/x-www-form-urlencoded will send the JSON payload as a form parameter called payload.
* Optionally, under "Secret", type a string to use as a secret key. You should choose a random string of text with high entropy. You can use the webhook secret to limit incoming requests to only those originating from GitHub. For more information, see [Validating webhook deliveries](https://docs.github.com/en/webhooks/using-webhooks/securing-your-webhooks).
* Under "Which events would you like to trigger this webhook?", select the webhook events that you want to receive. You should only subscribe to the webhook events that you need.
* If you chose Let me select individual events, select the events that you want to trigger the webhook.
* To make the webhook active immediately after adding the configuration, select Active.
* Click Add webhook.
* After you create a new webhook, GitHub will send you a simple ping event to let you know you've set up the webhook correctly. For more information, see [Webhook events and payloads](https://docs.github.com/en/webhooks/webhook-events-and-payloads#ping).

1. BASIC UNDERSTANDING OF GIT FILE



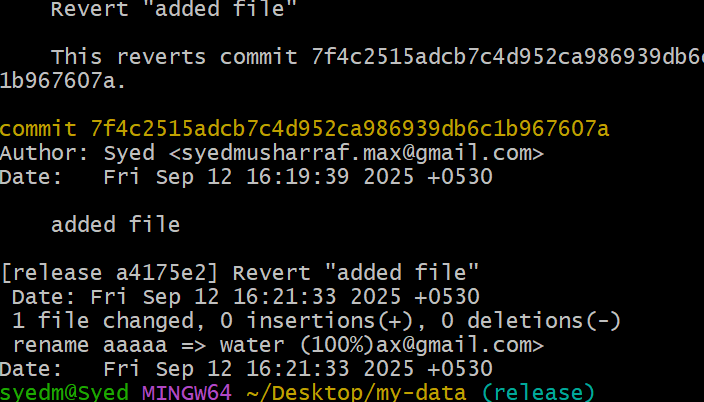
* The .git/ directory is the core component of a Git repository. It is a hidden directory located at the root of a Git-managed project and contains all the necessary information for Git to track changes, manage versions, and perform various operations. Essentially, it is where Git stores the entire history and metadata of your project.

1. Check all the logs of git.

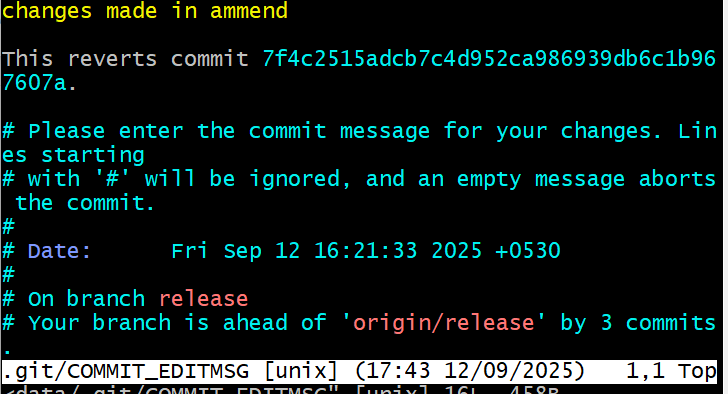


* Git log : to check all the logs

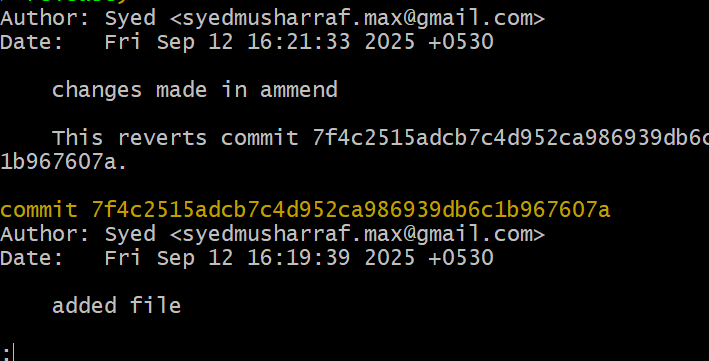
21. Rename the commit message.



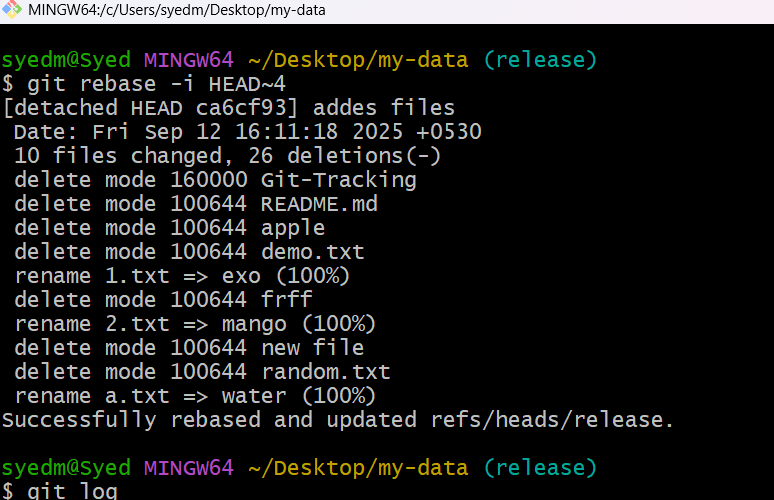
* Git commit - - amend : to change the commit messege and save it

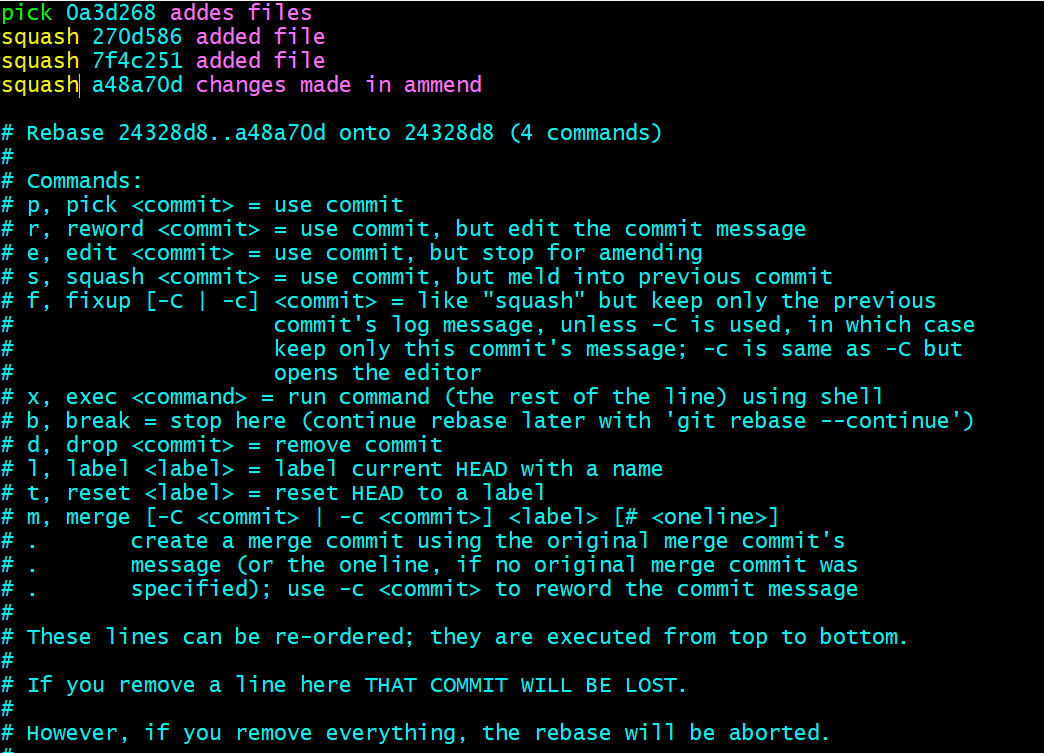


* You can see the commit messege has been changed.

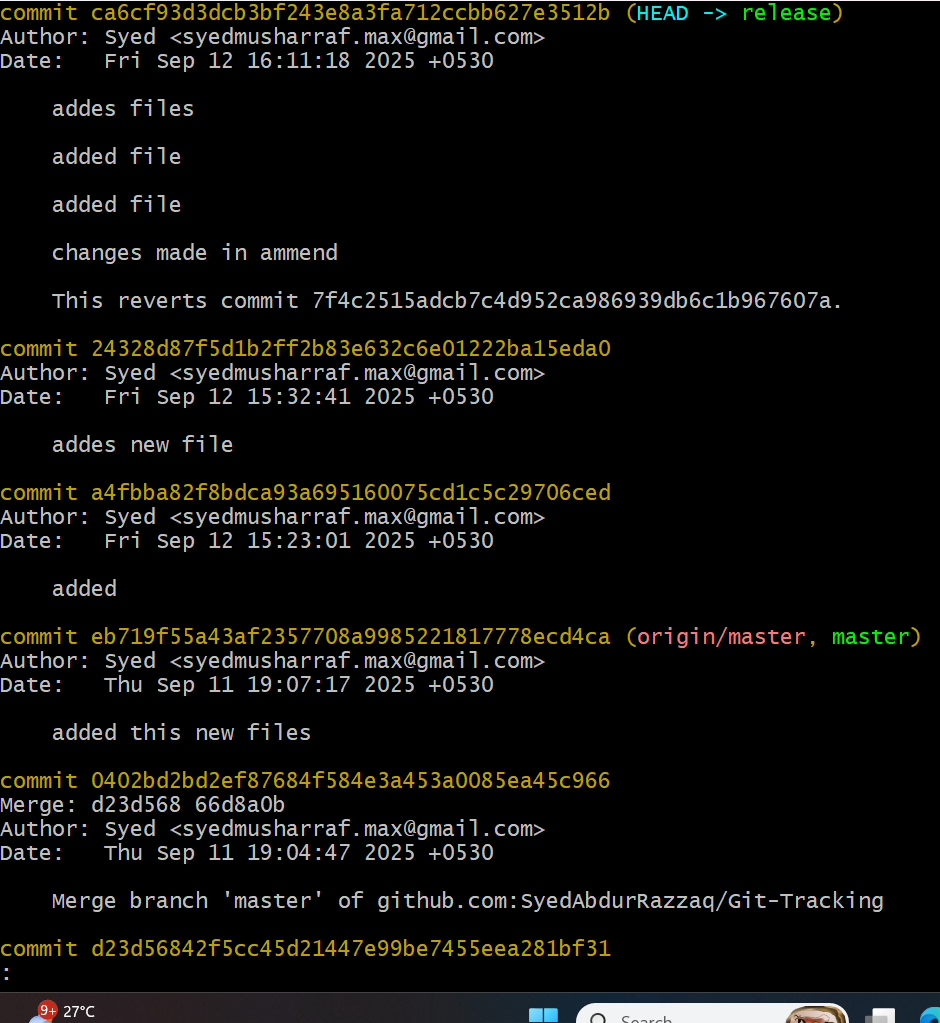


1. Merge multiple commits into single commit.





* Git rebase -i head~4 : to merge the commit



* Edit and add the messege an dsave it
* Git log you can see the commit has been merged.