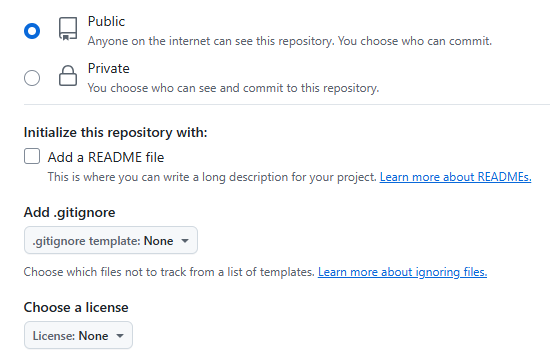
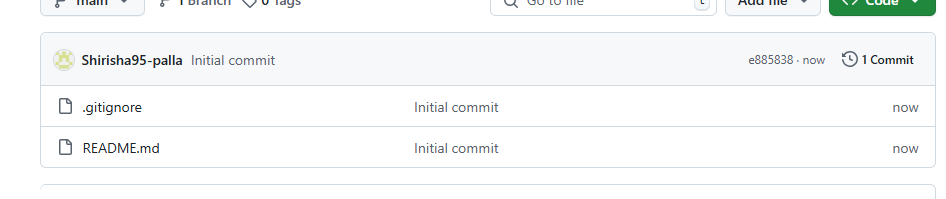
1. Install git.

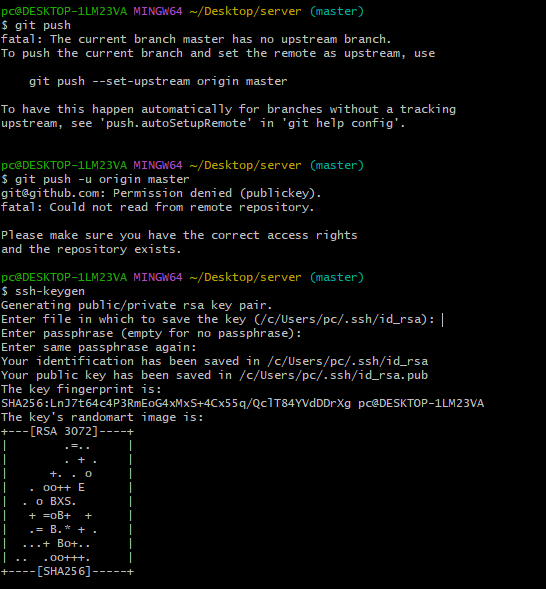


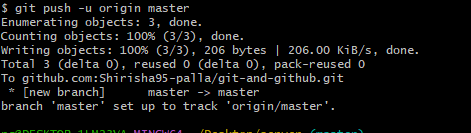
1. Create a repo in github with README.md and .ignore file.



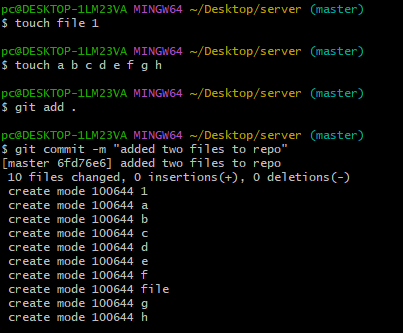


1. Clone the created repo to local.





1. Create two files in local repo.



1. Commit two files and push to central Repository.

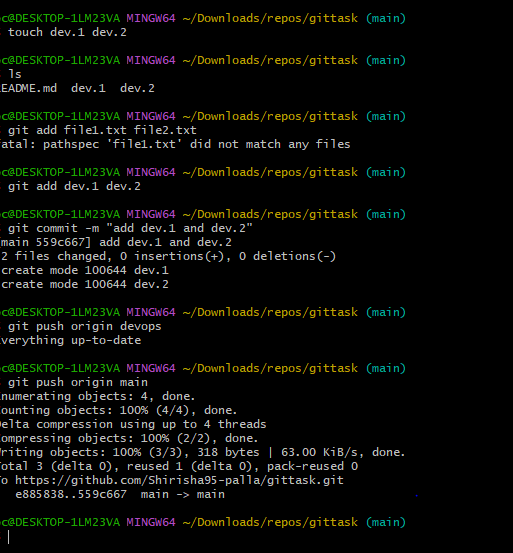
“git add .” - this command tells git to add everything in the current directory to

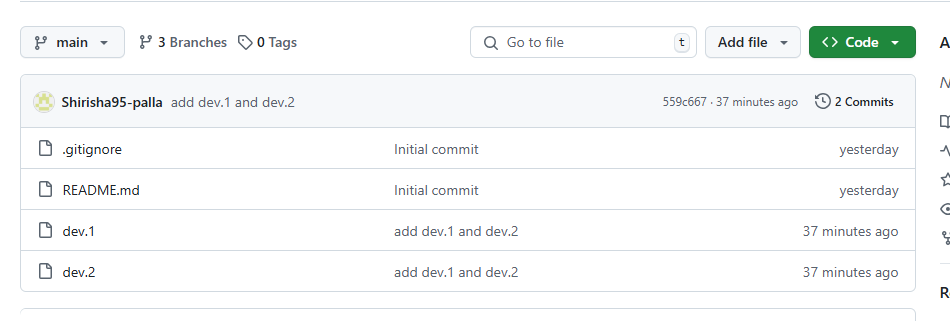
the staging area

● “git commit -m <your message>” - this command takes a snapshot/save of the

staged changes and lets you add a message, it creates a commit with

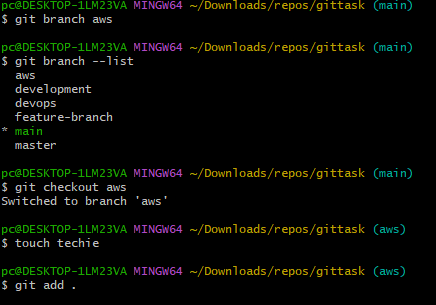
id,date,time,messsage,username and email id

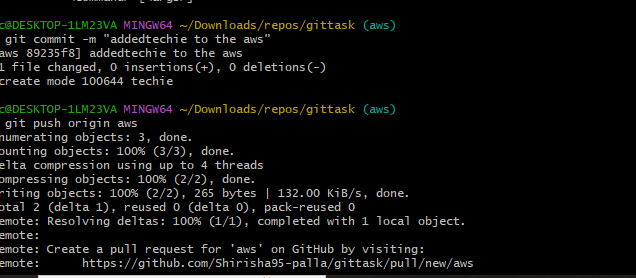


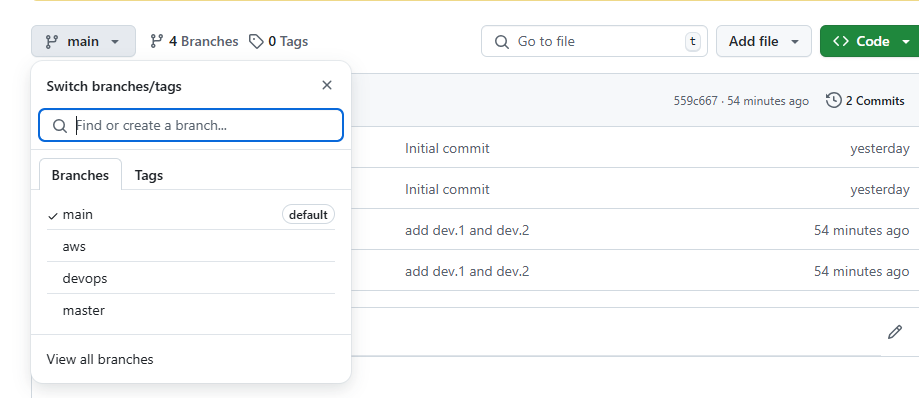


1. Create a branch in local and create a sample file and push to central.

● git branch <branch\_name> - makes an new branch by the name given  
● git checkout -b <branch\_name> and git switch -c <branch\_name> both can   
be used to make a branch and switch to that branch in one line.  
● git branch to see all the branches  
● rest of the steps are same as the above task(5).  
● Central refers to the remote repo like github

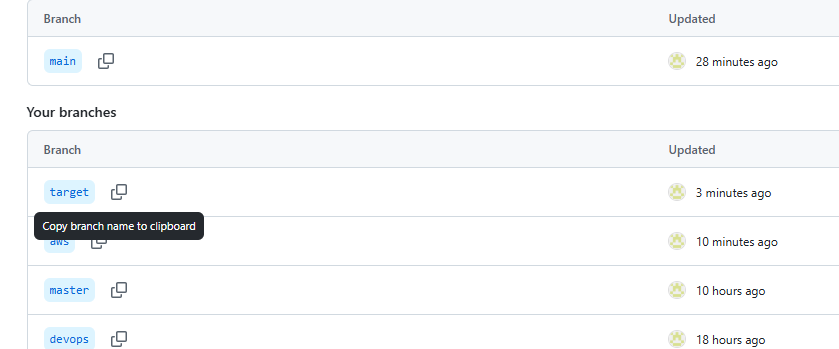


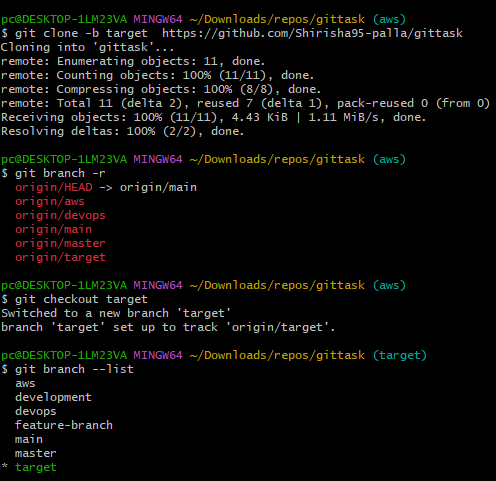


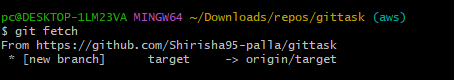


1. Create a branch in github and clone that to local.

● Go to your github and click on the main icon, in there just type the name of your   
new branch and then click on create branch  
● git fetch - this downloads all new data(branches, tags, commits) from the remote  
repository. Git fetch is different from git pull  
● git branch -r - to check the remote branches only  
● git clone -b <new\_branch> <github url> -  
(this step might not be necassary, could also just switch after fetching?)  
● git checkout <new\_branch> - checkout to the new branch  
● do git branch to see if the branch is present in the local

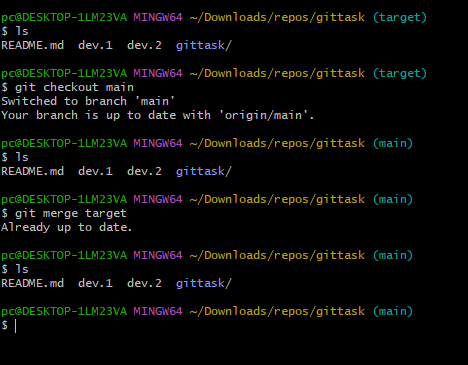






1. Merge the created branch with master in git local.

* Checkout to main branch
* then, git merge <target branch> to get the new files, codes, changes,commits   
  made on that branch to the branch you are in.

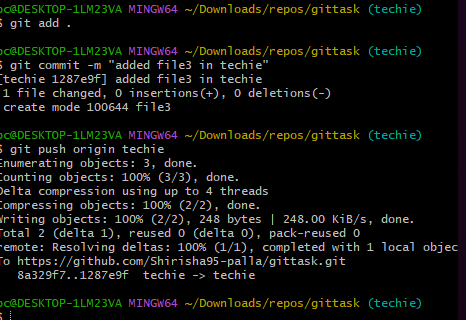


9) Merge the created branch with master in github by sending a pull request.

● make some changes to a branch and push it to main branch in remote repo using  
git push origin <branch\_name>  
● Go to github and you should see a ‘compare and pull request’ option pop up,   
click on it.  
● Make sure the base branch is main, add a title if you want and click on create   
pull request and then merge pull request

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

● **Create a new file in the branch and commit it. Then,  
● git push origin<current\_branch> - this sends the changes to the github repo**



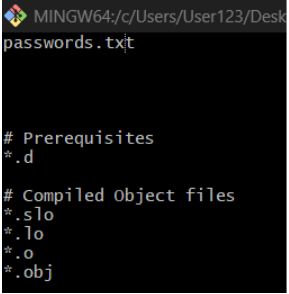


11) clone only a branch from github to local.

● First create a new branch in your remote repo  
● go to your local repo and do git fetch origin  
● then do git checkout -b <branch\_name> origin/<remote\_branch\_name> -   
this creates a local branch from origin/remotebranch and switches you to that   
branch  
● This wont clone the entire repo again, no new folders and wont cause a nested   
repo  
● git clone --branch <branch\_name> --single-branch <github url> - causes a nested repo

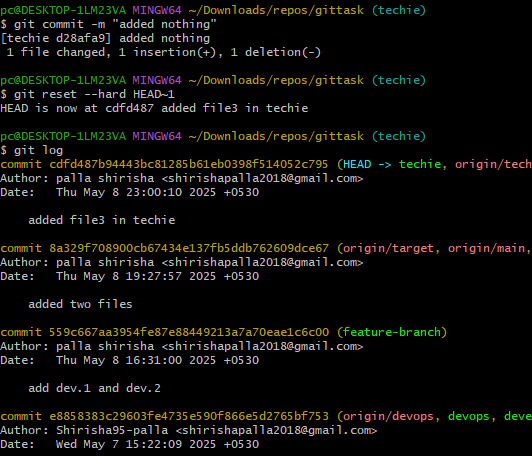
12)create a file with all passwords and make that untrackable with git.

Create a file with your passwords but dont commit it or add it.  
● open the .gitignore file and add the passwords file to it.  
● now commit only the .gitignore file and push it.  
● had to use git push origin remotelocalbranch:remotebranch1 because the   
local branch was not automatically tracking the remote one.  
● using regular git push origin <branch\_name> would not update the remote   
branch in this case and create a new branch on github

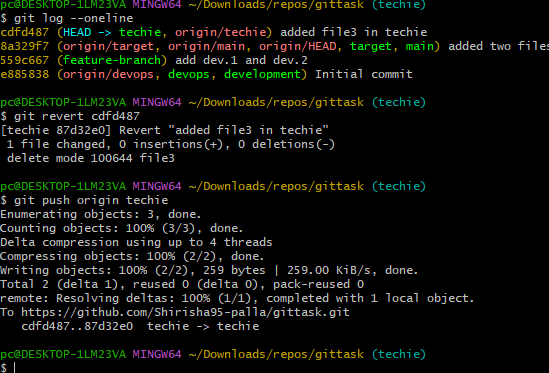


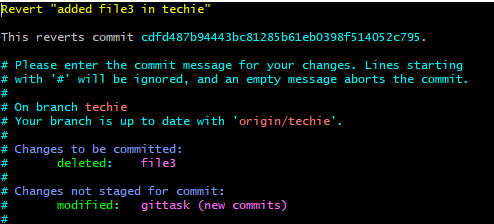
13) make a commit and make that commit reset without savings changes.

● git add . and git commit -m " " to make changes and add a new commit.  
● Reset the commit and discard the changes just made using -  
● git reset --hard HEAD~1 - this undos the last commit completely  
● git reset - Move the HEAD and current branch pointer to a different commit  
● hard - reset the working directory and staging area (dangerous)  
● HEAD~1 - Means "1 commit before HEAD" — the previous commit



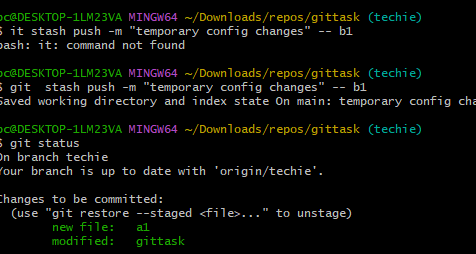
14) Revert a commited commit to the older version.



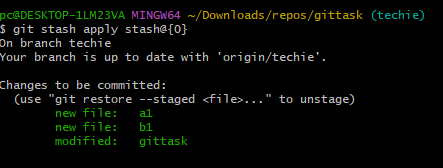




15) push a file to stash without savings the changes and work on another file.



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



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

● create a public key pair using ssh-keygen on the local system

● got to its location (~/.ssh) and copy the public key (id\_rsa)

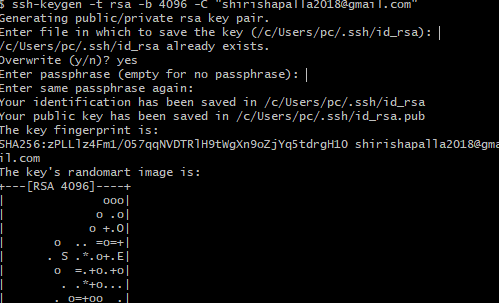
● Go to your GitHub profile and select SSH and GPG keys and click on new ssh

key

● Paste your public key in there and select Authentication Key as the type and click

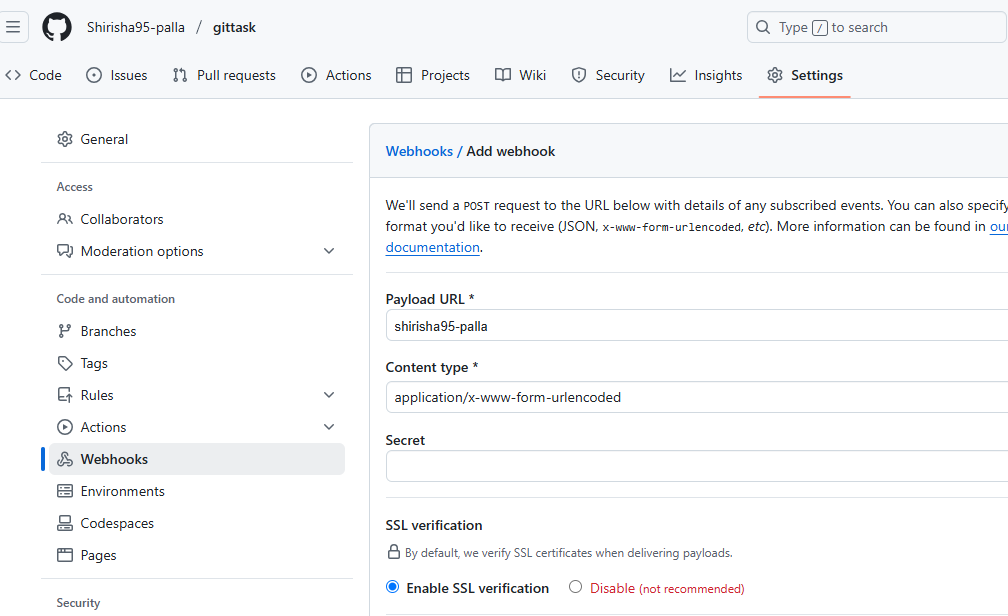
SSH key

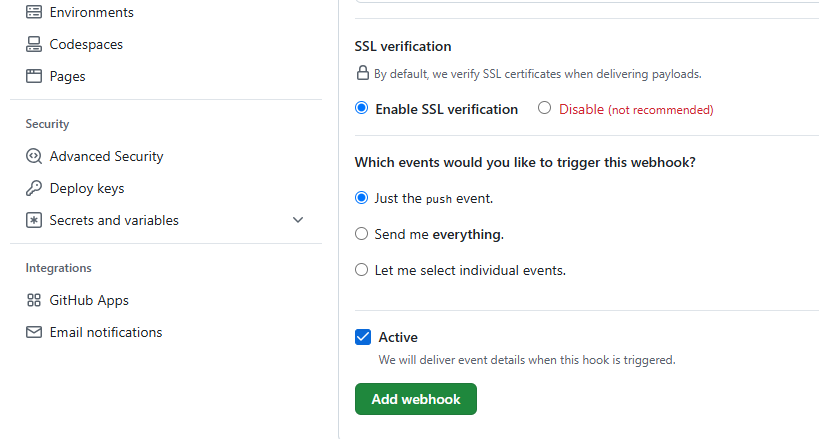
● To test the connection do ssh -T git@github.com followed by yes



18) configure webhooks to github.

● A webhook in GitHub lets you notify an external server or service whenever   
certain events happen in your repository — like a push, pull request, or release.  
● Go to your repository’s settings and click on webhooks on the left tab  
● Click on add webhooks





19) Basic understanding of .git file.

source - https://git-scm.com/book/en/v2/Git-Internals-Plumbing-and-Porcelain

● .git is a hidden folder that Git creates inside your repository when we run git init

or also when we run git clone <repo-url>.

● The .git folder is a hidden directory within a Git repository that stores all the

metadata necessary for version control, including the commit history, branch

information, and configuration settings. It essentially functions as a database for

your projet



● If you want to back up or clone your repository, copying this single directory

elsewhere gives you nearly everything you need.

● The description file is used only by the GitWeb program.

● The config file contains your project-specific configuration options, and the info

directory keeps a global exclude file for ignored patterns that you don’t want to

track in a .gitignore file.

● The hooks directory contains your client- or server-side hook scripts. Contains

sample scripts that run automatically when certain Git actions happen (like

commit, push, etc.)

● The objects directory stores all the content for your database,Stores everything

Git tracks like commits, files, folders (trees), tags, etc.

● The refs directory holds pointers to commit objects, including branch heads and

tags, which are used to track the history of the project. Contains references

(pointers) to commits. Eg:- heads/ → branches (like main, dev) or tags/ →

tags (like v1.0, release-1)

● the HEAD file points to the branch you currently have checked out, and the index

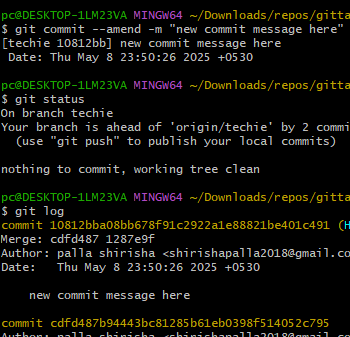
file is where Git stores your staging area information. ct's changes, allowing you to track, manage, and revert changes.

20) Check all the logs of git.



21) Rename the commit message

● git commit --amend - Git will open the last commit message in your text editor  
● Change the message, save and close the editor.  
● git commit --amend -m "new commit message" - to directly change it.



22)Merge multiple commits into single commit.  
● git rebase -i HEAD~3 - to squash the last 3 commits

● HEAD~3 to go back 3 commits from the current commit

● change the 2nd and 3rd pick to squas

